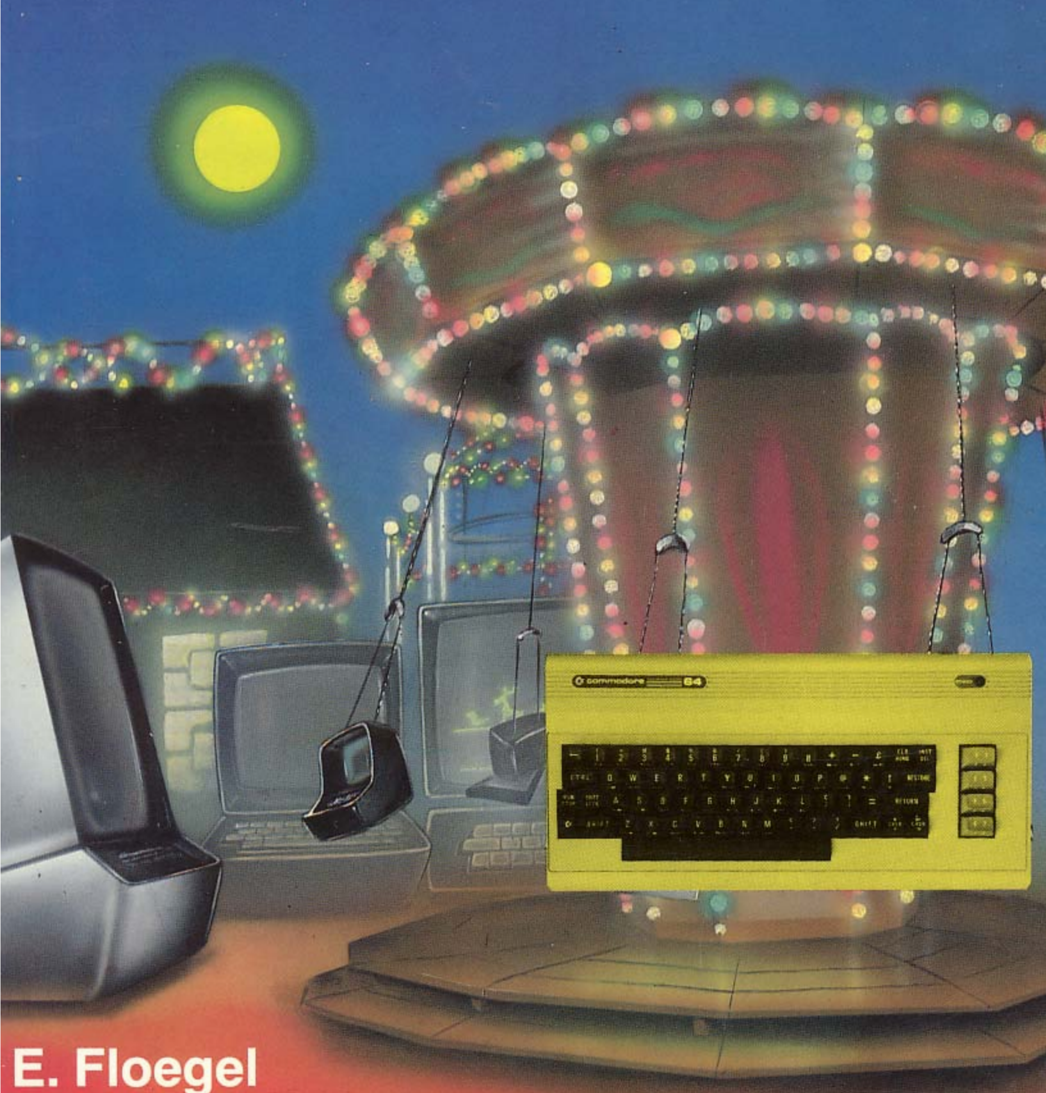
HARDWARE ERWEITERUNGEN
Für Commodore-64



E. Floegel

Verbindungen mit der Außenwelt ◆ Erweiterungen über den User- und Expansion Port ◆ Viele Programme

E. Floegel

HARDWARE ERWEITERUNGEN Für Commodore-64

Verbindungen mit der Außenwelt ◆ Erweiterungen über den User- und Expansion Port ◆ Viele Programme

Vorwort

Ein Rechner wird hauptsächlich zur Erfassung von Daten eingesetzt. In den meisten Fällen sind dies Daten, die aus dem wirtschaftlichen Bereich anfallen und über das Tastenfeld in den Rechner eingegeben werden.

Eine ganz andere Art von Daten sind solche, die durch Messungen gewonnen werden. Diese Daten aus einer analogen Welt müssen erst durch geeignete Wandler so umgeformt werden, daß sie von einem digitalen Rechner aufgenommen werden können.

Deshalb genügt zur Lösung solcher Messaufgaben nicht nur die Kenntnis des Rechners, sondern man braucht auch Kenntnisse über die Verarbeitung von analogen Signalen.

Die im Buch beschriebenen Messwerterfassungen sind immer nur als Beispiele zu betrachten. Jede reale Messaufgabe erfordert eine besondere Behandlung.

Der C-64 eignet sich besonders zum Einsatz in der Messtechnik. Über den USER-Port können direkt digitale Signale ein- und ausgegeben werden. Für Echtzeitanwendungen steht eine programmierbare Uhr und Timer zur Verfügung.

Ich wünsche allen Lesern einen guten Kontakt zur Aussenwelt.

Holzkirchen, Frühjahr 1984 Ekkehard Flögel

Inhaltsverzeichnis

1 Einführung

1. Einführung

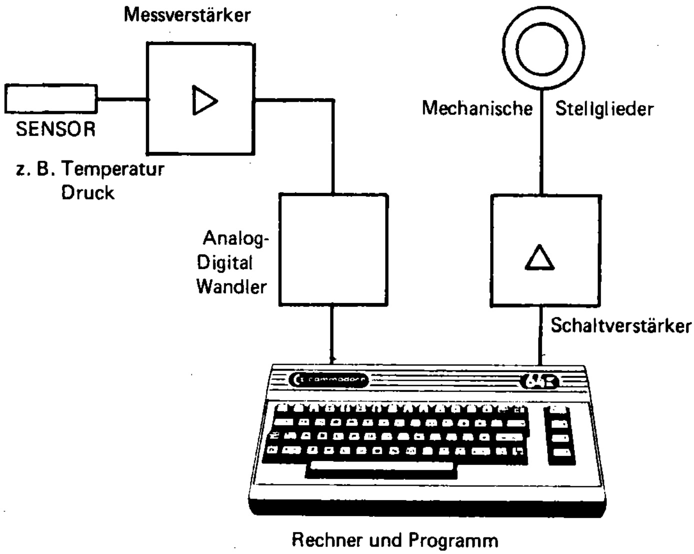
In den meisten Fällen wird ein Rechner zum Rechnen und zum Erfassen von Daten eingesetzt werden. Es gibt aber auch Fälle, in denen ein Rechner Steuerungsaufgaben übernehmen muß. Ein Beispiel ist die Erfassung der Temperatur. Wird die Innen- und Aussentemperatur gemessen, so kann eine Regelung der Heizung durchgeführt werden, die erheblich zur Senkung der Heizkosten beitragen kann. Solche Lösungen sind nur mit Hardware-Erweiterungen des Rechners möglich. Sensoren müssen an den Rechner angeschlossen werden. Durch die Auswertung der Signale durch das Programm werden dann Impulse an externe Geräte abgegeben und diese somit gesteuert.

Für die Lösung solcher Aufgaben sind weit mehr Kenntnisse notwendig als zur Programmierung eines Rechners. In Abbildung 1.1 ist eine Messdatenerfassung und Steuerung im Prinzip dargestellt.

Die Messung beginnt mit dem Sensor. Die erste Aufgabe ist es, den geeigneten Sensor für das zu messende Medium zu finden. Das Ausgangssignal wird meistens eine elektrische Spannung sein. Diese Signale kann ein Rechner nicht direkt verarbeiten. Ein Analog-Digital Wandler wandelt die Spannung in einen proportionalen Zahlenwert um. Dazu muß aber noch ein Messverstärker zwischen Sensor und Wandler geschaltet werden, um die Ausgangsspannung des Sensors an die Eingangsspannung des Wandlers anzupassen.

Im Rechner werden die vom Wandler gelieferten Zahlenwerte ausge-

wertet und entsprechende Ausgangssignale ausgegeben. Der Rechner gibt also über ein Tor ein Bitmuster aus, das über Schaltverstärker zum Beispiel mechanische Stellglieder ansteuert. Bei einer Heizungsregelung ist dies zum Beispiel ein Motor, der die Ölzufuhr drosselt oder aufmacht.



1.1 Messdatenerfassung und Steuerung mit einem Computer

Oft müssen solche Stellglieder über Relais oder OPTO-Koppler galvanisch getrennt werden, damit keine Störungen den Rechner beeinflussen.

Solche Aufgaben setzen also Grundkenntnisse in der Elektronik voraus. Sind diese nicht vorhanden, so sollte man besser bei der Datenverarbeitung oder bei Computerspielen bleiben.

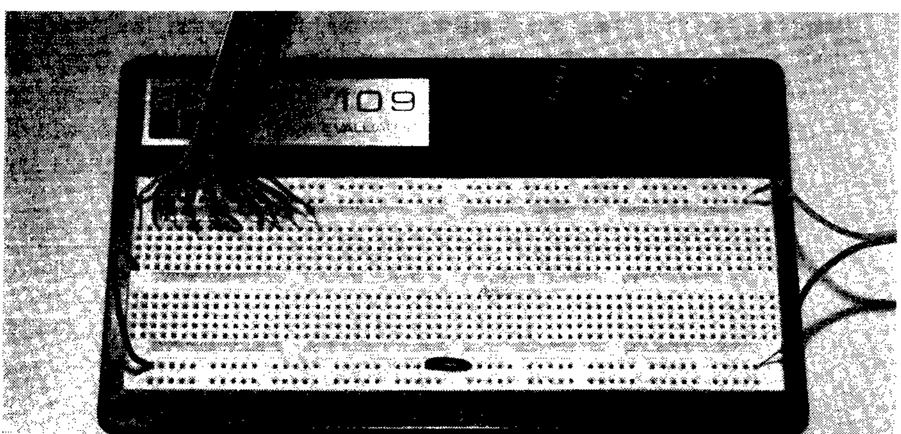
Ausserdem ist eine Grundausstattung an Hilfsgeräten notwendig. Abbildung 1.2 zeigt die Minimalausstattung.

Die Schaltung kann auf einem Steckbrett ohne zu Löten aufgebaut werden. Zur Überprüfung von Spannungen und Widerständen dient ein kleines Universal Messinstrument. Dies muß kein Präzisionsinstrument sein. Es genügt eine billige Ausführung, da im allgemeinen doch nur ja—nein Aussagen (Spannung da oder nicht, Durchgang vorhanden

oder nicht) gemessen werden. Zum Aufbau auf einem solchen Steckbrett benötigt man noch eine Flachzange zum Einstecken der Drähte oder Bauelemente, eine Pinzette und einen guten Seitenschneider. Eine Lupe kann zur Überprüfung der Schaltung und zur Entzifferung der Aufschriften von IC's oft gut bebraucht werden.



1.2 Hilfsmittel zum Aufbau einer Experimentierschaltung



1.3 Steckplatine

Abbildung 1.3 zeigt nochmals ein Steckbrett mit einem mehradrigen Kabel als Anschluß zum Rechner den Anschluß an ein Netzgerät.

Ein Netzgerät zur Versorgung der Experimentierplatine sollte folgende Spannungen und Ströme haben:

+5 Volt, 1 Ampere
+12 Volt, 0.5 Ampere
–12 Volt, 0.5 Ampere

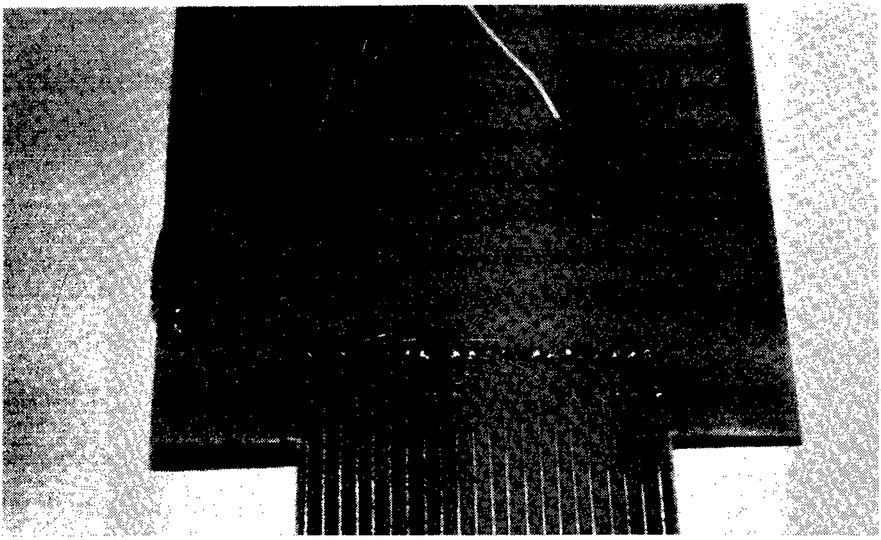
Mit der +5 Volt Spannung werden die TTL oder MOS Bausteine betrieben, während die ± 12 Volt zur Versorgung von Operationsverstärkern benutzt werden. Es wird dringend davor abgeraten, Netzgeräte mit einstellbarer Spannung zu verwenden. Durch Fehlbedienung können Bauteile oder sogar der Rechner zerstört werden.

Mit der Steckplatine können Schaltungen mit drei bis vier IC's gut aufgebaut werden. Sind in einer Schaltung mehr IC's notwendig oder soll eine Experimentierschaltung fest aufgebaut werden, so kann dies auf einer Lochrasterplatine in Fädeltechnik geschehen. Abbildung 1.4 zeigt die Rückseite einer Lochrasteplatine bei der die Verbindungen durch Fädeltechnik hergestellt wurden.

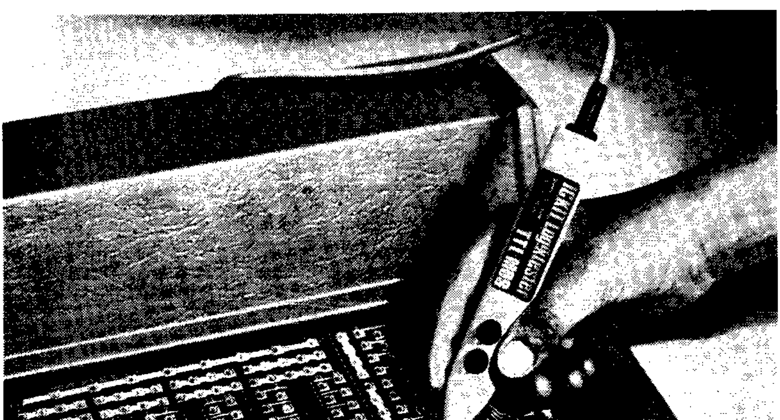
In Abbildung 1.2 ist zwischen dem Seitenschneider und der Pinzette ein Fädelgerät zu sehen. Ein lackisolierter Draht wird um die Beinchen eines IC's gewickelt und mit einem Lötkolben festgelötet. Dabei schmilzt die Lackisolierung und das Lötzinn verbindet den Anschluß mit dem Draht. Der Lötkolben sollte eine möglichst feine Spitze mit einer Temperatur von mindestens 220°C haben.

Es empfiehlt sich, für den Aufbau von Schaltungen temperaturgeregelte Lötkolben zu verwenden.

Diese minimale Grundausstattung kann nun beliebig erweitert werden. Eine erste Erweiterung ist ein Logiktester, wie er in Abbildung 1.5 gezeigt ist.



1.4 Fädeltechnik



1.5 Logiktester

Zwei Leuchtdioden zeigen den Pegelzustand eines Ausgangs an. Leuchtet die eine Diode, so ist der Ausgang H, leuchtet die andere Diode, so ist er L. Diese Bezeichnungen haben folgende Bedeutung:

H=1= Spannung zwischen 2.4 und 5 Volt,
L=0= Spannung zwsichen 0.8 und 2.2 Volt.

Dies ist die Definition von sogenannten TTL-Pegeln. In den folgenden Kapiteln werden beide Bezeichnungen (H oder Eins und L oder Null) benutzt.

Der Logiktester bietet noch einen weiteren Vorteil. Mit ihm können auch einzelne kurze Impulse angezeigt werden. Diese sind oft schwer mit einem Oszillographen zu messen. Ein Oszillograf gehört natürlich auch zu einer erweiterten Grundausstattung. Für anspruchsvollere Messungen sollte dies ein Zweistrahloszillograf sein, da damit Laufzeitmessungen zwischen Impulsen gemessen werden können. Die Bandbreite sollte mindestens 25 MHz betragen. Mit einer kleinsten Zeitauflösung von 0.1 μs.

Beim Umgang mit den CMOS-Bausteinen ist größte Vorsicht geboten. Vor dem Berühren eines Bauteils, auch innerhalb des Rechners sollte man sich durch Berühren einer Masseleitung elektrostatisch entladen. Dazu befestigt man an einer Masseleitung einen Widerstand von 1M. Wenn man diesen berührt, so wird der Körper entladen, ohne daß dabei kleine elektrische Schläge auftreten. Ferner sollten niemals Stecker in den USER-Port oder Expansion Port gesteckt werden ohne daß der Rechner vorher abgeschaltet war.

Nach diesen allgemeinen Bemerkungen zu der Hardware nun einige Anmerkungen zu der Programmiersprache.

BASIC ist die am meisten verbreitete Programmiersprache. Sie hat bei der Anwendung für Steuer- und Regelungsaufgaben den Nachteil, daß sie in vielen Fällen zu langsam ist. Es empfiehlt sich daher auf Assemblerprogrammierung umzusteigen. In dieser Sprache ist der Rechner meist schneller als externe Geräte, so daß unter Umständen Warteschleifen programmiert werden müssen.

Eine ganz andere Sprache ist FORTH. Sie ist langsamer als Assembler, ist aber wesentlich schneller als BASIC. Sie besitzt aber gegenüber diesen Sprachen einen ganz entscheidenden Vorteil, der in der Struktur dieser Sprache liegt. In BASIC und Assembler sind die Befehle zu Programmteilen und Unterprogrammen zusammengefasst. Diese müssen zu einem Programm zusammengesetzt werden, das mit einem Startbefehl, in BASIC ist es der Befehl RUN, gestartet wird. FORTH verhält sich ganz anders. Hier sind die Programmteile Worte, die in einem Wörterbuch gespeichert sind. Das Starten eines Programms in FORTH bedeutet den Aufruf eines Wortes. Sind die Befehle, die dieses Wort enthält, abgearbeitet, so wartet das Programm auf die Eingabe eines neuen Wortes. So ist es zum Beispiel möglich, durch die Eingabe von Worten eine Anlage zu steuern. Folgende Eingaben sind denkbar:

MOTOR 2 EIN

VENTIL 5 LANGSAM ZU

Das Wort MOTOR wählt das Tor mit den Steuerleitungen zu den Motoren aus. Die Eingabe 2 EIN gibt einen Einschaltimpuls auf die Leitung zum Motor 2. Ein Beispiel für diese Programmierung ist in Kapitel 2 zu finden. Das Wort LANGSAM im vorherigen Beispiel verlängert eine Warteschleife.

FORTH wird zum Beispiel als Sprache zur Steuerung von Robotern in den Trickfilmstudios von Hollywood verwendet. Anstatt Objekte zu bewegen, werden Filmkameras auf Roboter montiert und diese bewegt. Auf diese Weise sind die Trickaufnahmen zum Film "Krieg der Sterne" entstanden.

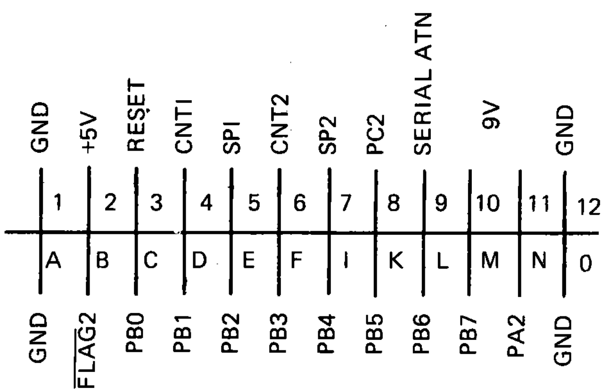
Bei den meisten Beispielen ist die Programmierung in den drei Sprachen BASIC, ASSEMBLER und FORTH angegeben.

Notizen

2 Hardware Erweiterungen über den USER-Port

2.0 Hardware-Erweiterungen über den USER-Port

Der USER-Port des C-64 kann für Hardware-Erweiterungen verwendet werden. Die an diesem Tor verfügbaren Leitungen sind mit einem CIA 6526 verbunden. Die Steckerbelegung zeigt Abbildung 2.1.



2.1 Anschlußbelegung des USER-Ports

Eine Beschreibung der einzelnen Anschlüsse erfolgt bei der Beschreibung des Bausteins CIA 6526.

Wie schon erwähnt, werden die Programme in den drei Programmiersprachen BASIC, ASSEMBLER und FORTH angegeben. Folgende Vereinbarungen sind für die Variablennamen getroffen worden:

BASIC:

```
10 A=56576
20 PB=A+1
30 DB=A+3
40 L1=A+4
50 H1=A+5
60 L2=A+6
70 H2=A+7
80 CA=A+14
90 CB=A+15
```

ASSEMBLER:

```
PORTB     EQU  $DD01
DDRB      EQU  $DD03
T1        EQU  $DD04
T2        EQU  $DD06
CRA       EQU  $DD0E
CRB       EQU  $DD0F
```

FORTH:

```
SCR # 10
( I/O                           9.11.  EF)
HEX
DD01 CONSTANT PORTB
DD03 CONSTANT DDRB
DD04 CONSTANT T1
DD06 CONSTANT T2
DD0E CONSTANT CRA
DD0F CONSTANT CRB
DECIMAL
```

2.2 Vereinbarungen

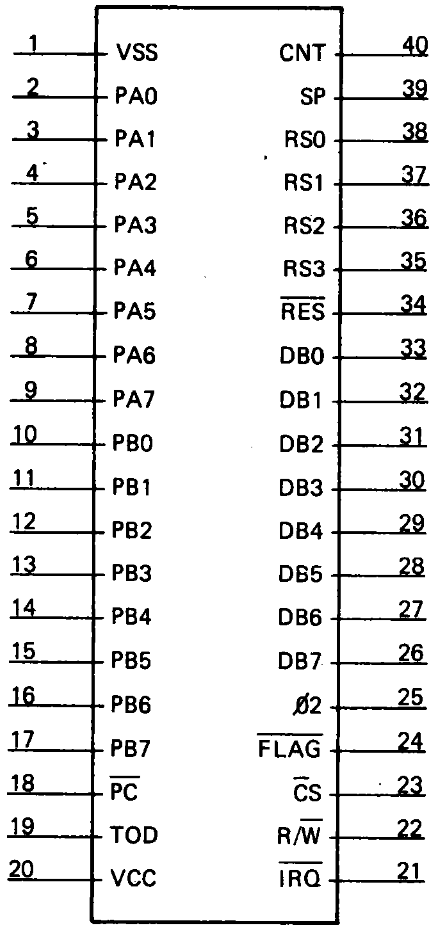
2.1 Der Baustein CIA 6526

Die Abkürzung CIA bedeutet Complex Interface Adapter. Dieser Baustein enthält:

— 16 bidirektionale I/O Leitungen

– 8 oder 16 Bit Handshaking für Lesen oder Schreiben
– 2 unabhängige Zähler
– eine 24-Stunden Uhr
– ein 8-Bit Schieberegister für serielle Ausgabe

Die Abbildung 2.3 zeigt die Pinbelegung des Bausteins CIA 6526.



2.3 Pinbelegung des CIA 6526

Die interne Adressierung und die Registerauswahl geschieht durch die Leitungen $\overline{RES}$, R/$\overline{W}$, $\overline{CS}$, Ø2, RS3, RS2, RS1 und RS0. Mit den letzten vier Leitungen werden die Register des 6526 ausgewählt. Diese sind in der Abbildung 2.4 angegeben. Der C-64 enthält zwei 6526. Die in der Abbildung angegebenen Adressen sind die Adressen des zweiten 6526. Von diesem sind am USER-Port die Datenleitungen PB0 bis PB7 und PA2 angeschlossen.

Adresse	Name	Bezeichnung
DD00	PORTA	I/O TOR A
DD01	PORTB	I/O TOR B
DD02	DDRA	Datenrichtungs Reg. A
DD03	DDRB	Datenrichtungs Reg. B
DD04	T1L	Timer 1 LO-BYTE
DD05	T1H	Timer 1 HI-BYTE
DD06	T1L	Timer 2 LO-BYTE
DD07	T2H	Timer 2 HI-BYTE
DD08	DOD10	Tagesuhr 1/10 Sec.
DD09	TODS	Tagesuhr Sekunden
DD0A	TODM	Tagesuhr Minuten
DD0B	TODH	Tagesuhr Stunden, AM/PM
DD0C	SDR	Serielles Daten Reg.
DD0D	ICR	Interrupt CTRL-Register
DD0E	CRA	Control-Reg. A
DD0F	CRB	Control-Reg. B

2.4 Register des CIA 6526

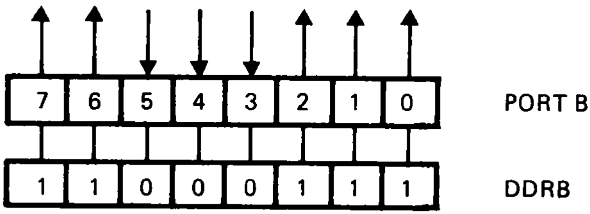
2.11 Programmierung der Tore

Zu jedem Tor ist ein Daten-Richtungsregister zugeordnet. Zum Tor A (PORTA) gehört das Richtungsregister DDRA, zum Tor B das Richtungsregister DDRB. Jeder Ausgangsleitung eines Tores ist ein Bit des Richtungsregisters zugeordnet. Ist dieses Bit Eins, so ist diese Leitung eine Ausgangsleitung, ist es Null, so ist diese Leitung ein Eingang. Ein Beispiel zeigt Abbildung 2.5.

Durch das Bitmuster 11000111 = $C7 = 199 sind die Leitungen PB0 bis PB2, PB6 und PB7 Ausgänge und die Leitungen PB3 bis PB5 Ein-

gänge.

Am USER-Port sind die Leitungen $\overline{\text{PC}}$ und $\overline{\text{FLAG}}$ des zweiten 6526 angeschlossen. Diese Leitungen können für den Datenaustausch mit Handshaking verwendet werden. Die Leitung $\overline{\text{PC}}$ wird einen Taktzyklus lang negativ, wenn ein Schreib- oder Lesebefehl auf Tor B erfolgt. Die Leitung $\overline{\text{FLAG}}$ ist ein Eingang. Eine negative Flanke setzt im Interrupt Register das FLAG-Bit.



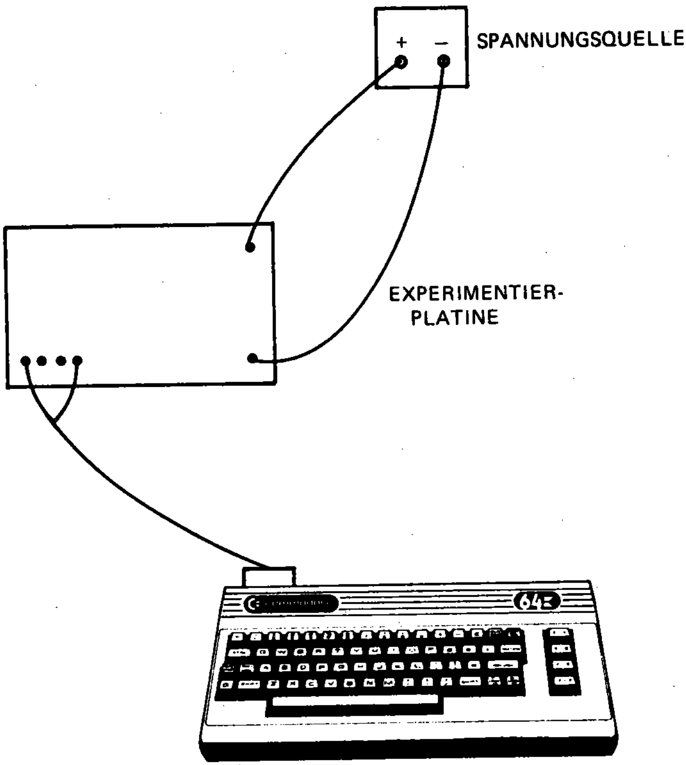
2.5 Zuordnung von Tor und Richtungsregister

2.12 Ein- und Ausschalten von Verbrauchern

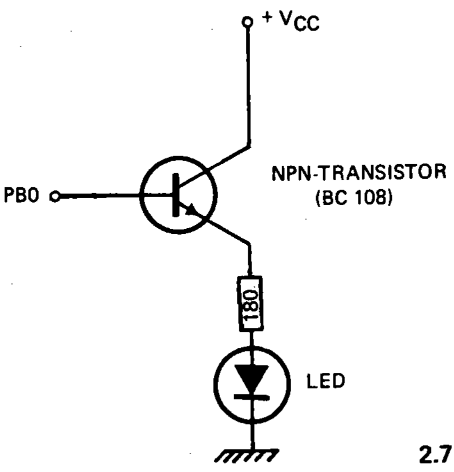
2.121 Schalten einer Leuchtdiode

Die Leitungen von Tor B werden zum Ein- und Ausschalten von Verbrauchern benutzt. Dazu wird der in Abbildung 2.6 gezeigte Versuchsaufbau verwendet. Die Schaltungen werden auf einer Experimentierplatte aufgebaut. Diese wird von einer externen Spannungsquelle versorgt. Über ein mehradriges Kabel ist die Experimentierplatine mit dem USER-Port verbunden.

Wenn die Leitungen des USER-Ports als Ausgänge geschaltet sind, so können diese mit 2 TTL-Lasten belastet werden. Dies entspricht einem Strom von wenigen Milliampere. Benötigt der Verbraucher mehr Leistung zum Schalten, so müssen Transistoren oder Relais zwischengeschaltet werden. Abbildung 2.7 zeigt die Ansteuerung einer Leuchtdiode. Der NPN Transistor wird in Emitterschaltung betrieben. Vor die Leuchtdiode ist ein Widerstand von 180 Ohm zur Strombegrenzung in Serie geschaltet.



2.6 Versuchsaufbau



2.7 Ansteuerung einer Leutdiode

In dieser Schaltung leuchtet die Diode, wenn an PBO eine Eins ausgegeben wird. Abbildung 2.8 zeigt das Programm zum Ein- und Ausschalten.

BASIC:

```
POKE DB,1
POKE PB,1
POKE PB,0
```

ASSEMBLER:

```
                        ORG $C000
C000: A901   INIT       LDA #1
C002: 8D03DD            STA DDRB ; PB0 AUSGANG
C005: 00                BRK

C006: A901   EIN        LDA #1
C008: 8D01DD            STA PORTB ; LED EIN
C00B: 00                BRK

C00C: A900   AUS        LDA #0
C00E: 8D01DD            STA PORTB ; LED AUS
C011: 00                BRK
```

FORTH:

```
SCR # 19
  0 ( EIN- UND AUSSCHALTEN  9.11.EF)
  1 1 DDRB C!
  2 : EIN 1 PORTB C! ;
  3 : AUS 0 PORTB C! ;
  4
  5 ;S
```

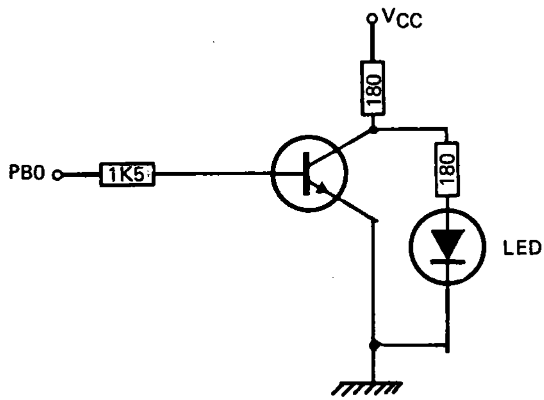
2.8 Ein- und Ausschalten einer Leuchtdioden

In BASIC wird durch den POKE Befehl POKE DB,1 das Datenrichtungsregister gesetzt. Mit POKE PB,1 wird die Leuchtdiode eingeschaltet, mit POKE PB,0 wieder ausgeschaltet.

In Assembler sind drei kleine Programmteile vereinbart. Ab Adresse C000 wird PB0 zum Ausgang gemacht. Ab Adresse C006 die Leuchtdiode ein-, und ab Adresse C00C wieder ausgeschaltet.

Zum Tor A In FORTH wird beim Compilieren das Datenrichtungsregister gesetzt. Das Wort EIN schaltet die Diode ein, das Wort AUS schaltet sie wieder aus.

In Abbildung 2.9 ist die Ansteuerung der Leuchtdiode in eine Kollektorschaltung geändert worden. Nun wird die Leuchtdiode durch eine Null an PB0 ein- und durch eine Eins ausgeschaltet.



2.9 Geänderte Ansteuerung der Leuchtdiode

Die in Abbildung 2.8 angegebenen Werte zum Ein- und Ausschalten werden daher vertauscht.

2.122 Ansteuerung von mehreren Leuchtdioden

Für die folgenden Programme sind 8 Leuchtdioden mit der in Abbildung 2.7 angegebenen Schaltung mit den Leitungen PB0 bis PB7 verbunden. Damit sollen vier Beispiele, Ausgabe eines Bitmusters, Einschalten einer bestimmten Leuchtdiode, Lauflicht und Lichtbalken gezeigt werden.

Ausgabe eines Bitmusters

Das Programm zeigt Abbildung 2.10.

BASIC:

```
100 POKE DB,255
110 INPUT"N=";N
120 IF N<0 THEN END
130 POKE PB,N
140 GOTO 110
```

ASSEMBLER: nicht implementiert

FORTH:

```
SCR # 12
  0 ( I/O PORTB                    10.11.EF)
  1 HEX
  2 FF DDRB C!
  3
  4 : AUS ( N) PORTB C! ;
  5
```

2.10 Ausgabe eines Bitmusters

In BASIC werden in Zeile 100 alle Leitungen zu Ausgängen gemacht. In einer Schleife wird eine Zahl eingegeben, von welcher die untersten acht Bit als Bitmuster ausgegeben werden. Diese Schleife wird solange durchlaufen, bis eine Zahl kleiner Null eingegeben wird.

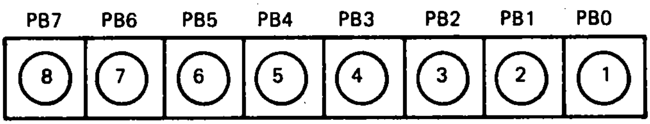
Ein Assemblerprogramm wäre sehr umfangreich geworden, da über die Tastatureingabe erst aus den ASCII-Zeichen eine Zahl hätte gebildet werden müssen. Deshalb wird auf ein Assemblerprogramm verzichtet.

In FORTH wird das Wort AUS definiert. Dieses holt eine Zahl vom Stapel und gibt sie and das Tor B aus.

Ein- und Ausschalten einer bestimmten Leuchtdiode

Die acht Leuchtdioden sind, wie in Abbildung 2.11 gezeigt, von rechts nach links von eins bis acht durchnummeriert. Der Ausgang PB0 steuert Leuchtdiode Eins, der Ausgang PB7 die Diode Acht. Durch 4 EIN soll

nun die Leuchtdiode Vier eingeschaltet werden, ohne daß die anderen Dioden beeinflußt werden. Abbildung 2.12 zeigt das Programm.



2.11 Acht Leuchtdioden in einer Reihe

BASIC:

```
100 POKE DB,255
110 S=0
120 POKE PB,S
130 INPUT  "WELCHE LED ";N
135 IF N<0 THEN END
140 N=N-1: A=2^N
150 INPUT"E)IN ODER A)US ";A$
160 IF LEFT$(A$,1)="E" THEN 200
170 IF LEFT$(A$,1)="A" THEN 220
180 GOTO 150
200 S=S+A: IF S<256 THEN POKE PB,S
210 GOTO 130
220 S=S-A: IF S>-1 THEN POKE PB,S
230 GOTO 130
```

ASSEMBLER: nicht implementiert

FORTH:

```
: PINIT ( -N) FF DDRB C! 0 DUP
  PORTB C! ;
: NEW ( N-N') 1 SWAP DUP 1 = IF
  DROP ELSE 1 DO 2 * LOOP THEN ;

: EIN ( NN'-N") NEW OR DUP PORTB
  C! ;
: AUS ( NN'-N") NEW XOR DUP
  PORTB C! ;

```

2.12 Schalten bestimmter Leuchtdioden

Im BASIC Programm wird das Datenrichtungsregister gesetzt und die Variable S Null gesetzt. Dieser Wert wird auch an das Tor ausgegeben. In S ist der Zustand der einzelnen Leuchtdioden gespeichert. In Zeile 130 wird die Nummer der Leuchtdiode eingegeben und in Zeile 150, ob sie ein- oder ausgeschaltet werden soll. Der Programmablauf wird am besten an einem Beispiel deutlich gemacht.

Bis jetzt sei Diode 5 eingeschaltet. Damit ist S = 16. Nun soll die Leuchtdiode 8 dazugeschaltet werden. Für N wird 8 eingegeben. In Zeile 140 wird N=7 und A=2 ↑ 7 = 128. In Zeile 200 wird A zu S addiert und der neue Wert von S, wenn er kleiner als 256 ist, an das Tor ausgegeben. In dem Beispiel ist S = 144. Damit sind die Leuchtdioden 5 und 8 eingeschaltet. Beim Ausschalten wird der Wert von A und S abgezogen. Das Programm ist in dieser Form nicht fehlerfrei. Wird eine bereits eingeschaltete Leuchtdiode nochmals eingeschaltet, so werden andere Dioden davon beeinflußt. Es wird nur geprüft, ob S nicht größer 255 oder kleiner Null ist. Wird N negativ eingegeben, so wird das Programm beendet.

Auf ein Assemblerprogramm wird auch hier verzichtet.

Das FORTH-Programm arbeitet ähnlich. Das Wort PINIT setzt das Datenrichtungs-Register, legt auf dem Stapel eine Null ab und gibt diese an das Tor aus. Das Wort NEW bestimmt das Bit, das gesetzt werden muß, um eine Leuchtdiode zu schalten. Wenn N 〉 1 ist wird N—1 mal mit Zwei multipliziert. Wird das Wort EIN aufgerufen, so ist vor der Ausführung der alte Zustand und die Nummer der Leuchtdiode auf dem Stapel. Nach der Ausführung ist der neue Zustand auf dem Stapel. Das Einschalten einer Leuchtdiode geschieht mit der ODER Funktion (OR), das Ausschalten mit der EXCLUSIV-ODER Funktion. Die Wortfolge

```
PINIT
3 EIN
```

schaltet Leuchtdiode 3 ein. Danach können mit 5 EIN, 3 AUS usw. andere Dioden geschaltet werden. Am Schluß, wenn keine weiteren Dioden mehr geschaltet werden sollen, muß mit DROP das oberste Element des Stapels beseitigt werden.

Lauflicht

Im Programm in Abbildung 2.13 werden die Leuchtdioden nacheinander ein- und wieder ausgeschaltet. Dadurch entsteht der Eindruck eines laufenden Lichtes.

BASIC:

```
100 POKE DB,255
110 POKE PB,0
120 A=1
130 POKE PB,A
140 GOSUB 200
150 A=A*2
160 IF A=256 THEN A=1
170 GOTO 130
200 FOR I=1 TO 50
210 NEXT I: RETURN
```

ASSEMBLER:

```
                             ORG $C000
C000: A9FF                   LDA #$FF
C002: 8D03DD                 STA DDRB
C005: A900                   LDA #00
C007: 8D01DD                 STA PORTB
C00A: 38                     SEC
C00B: 2A        M            ROL
C00C: 2016C0                 JSR WAIT
C00F: 8D01DD                 STA PORTB
C012: 90F7                   BCC M
C014: B0F5                   BCS M

C016: A280      WAIT         LDX #$80
C018: A0FF                   LDY #$FF
C01A: 88        W            DEY
C01B: D0FD                   BNE W
C01D: CA                     DEX
C01E: D0FA                   BNE W
C020: 60                     RTS
```

FORTH:

```
SCR # 13
( I/O PORTB                 10.11.EF)
HEX
FF DDRB C!
DECIMAL
: WAIT ( N) 0 DO LOOP ;
: LL 1 BEGIN DUP PORTB C! 2 *
  DUP 256 = IF DROP 1 THEN
  2000 WAIT ?TERMINAL UNTIL ;


;S
```

2.13 Lauflicht

Im BASIC-Programm ist der Wert der Variablen A zu Anfang Eins. Dieser Wert wird jeweils mit Zwei multipliziert und an das Tor B ausgegeben. Wenn A den Wert 256 erreicht hat, wird dieser auf Eins zurückgesetzt. Das Unterprogramm ab Zeile 200 ist eine Verzögerungsschleife. Das Programm muß mit RESTORE STOP beendet werden.

Im Assembler Programm wird das Tor B auf Null gesetzt. Danach wird das CARRY-Bit gesetzt und mit ROL jeweils eine Stelle nach links geschoben und an das Tor ausgegeben.

Das FORTH-Programm verwendet den gleichen Algorithmus wie das BASIC-Programm. Die oberste Zahl auf dem Stapel wird immer mit Zwei multipliziert. Wird der Wert 256 erreicht, so wird wieder eine Eins auf den Stapel gelegt. Das Wort WAIT stellt auch hier eine Warteschleife dar.

Lichtbalken

Im Programm in Abbildung 2.14 wird ein Lichtbalken simuliert. Die einzelnen Leuchtdioden werden nacheinander solange eingeschaltet, bis alle leuchten. Danach werden alle auf einmal ausgeschaltet und der Zyklus beginnt aufs Neue.

Das Programm entspricht im wesentlichen dem Programm in Abbildung 2.13. Im BASIC Programm wird A immer mit zwei multipliziert und zu B addiert.

Im Assemblerprogramm wird in jeder Schleife das CARRY-Bit gesetzt und in den Akkumulator geschoben.

BASIC:

```
100 POKE DB,255
110 POKE PB,0
120 A=1:B=1
130 POKE PB,B
140 GOSUB 200
150 A=A*2:B=B+A
160 IF A=256 THEN 120
170 GOTO 130
200 FOR I=1 TO 50
210 NEXT I: RETURN
```

ASSEMBLER:

```
                               ORG $C000
C000: A9FF                     LDA #$FF
C002: 8D03DD                   STA DDRB
C005: A900      M              LDA #00
C007: 8D01DD                   STA PORTB
C00A: 38        M1             SEC
C00B: 2A                       ROL
C00C: 2016C0                   JSR WAIT
C00F: 8D01DD                   STA PORTB
C012: 90F6                     BCC M1
C014: B0EF                     BCS M

C016: A280      WAIT           LDX #$80
C018: A0FF                     LDY #$FF
C01A: 88        W              DEY
C01B: D0FD                     BNE W
C01D: CA                       DEX
C01E: D0FA                     BNE W
C020: 60                       RTS
```

FORTH:

```
SCR # 14
( I/O PORTB                    10.11.EF)
HEX
FF DDRB C!
DECIMAL

: LB 0 BEGIN DUP PORTB C! 2 * 1+
  DUP 255 > IF DROP 0 THEN 1000
  WAIT ?TERMINAL UNTIL ;

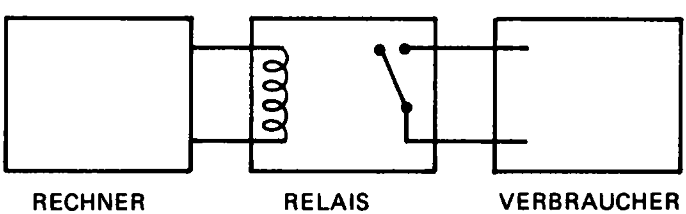

;S
```

2.14 Lichtbalken

Im FORTH-Programm wird die Zahlenfolge 0, 1, 3, 7, 15 usw. bis 255 erzeugt und an das Tor ausgegeben.

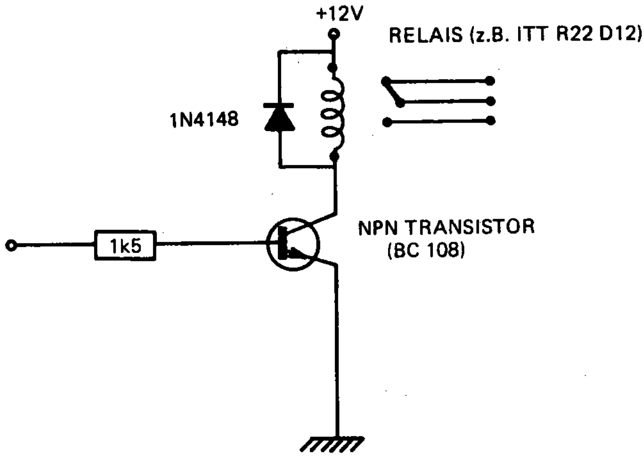
2.123 Ansteuerung von Relais

Zum langsamen Schalten von großen Lasten und zur galvanischen Trennung von Rechner und Verbraucher werden oft Relais verwendet. Die Abbildung 2.15 zeigt ein Prinzipschaltbild. Der Rechner steuert über einen Verstärker die Spule eines Relais. Der davon galvanisch getrennte Kontakt schaltet die Last eines Verbrauchers.



2.15 Galvanische Trennung von Rechner und Verbraucher

Eine aufgebaute Schaltung zeigt Abbildung 2.16. Vom Rechner wird über eine Transistorstufe ein Reed-Relais geschaltet. Die Schaltspannung dieses Relais beträgt 12 Volt. Reed-Relais haben eine kleine Leistungsaufnahme und können, bedingt durch ihren Aufbau, kleine und mittlere Ströme schnell schalten. Über die Spule des Relais ist in Sperrichtung eine Schutzdiode geschaltet. Diese schließt die beim Abschalten der Spannung auftretenden induktiven Spannungsspitzen kurz, und verhindert so eine Zerstörung des Transistors.

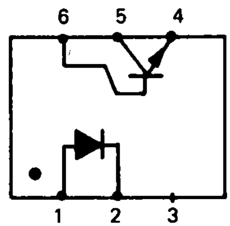


2.16 Schalten eines Relais

In den meisten Fällen reicht bei kleinen Steuerleistungen eine Transistorstufe aus. Für höhere Spannungen und Steuerleistungen verwendet man am besten Darligton Transistoren.

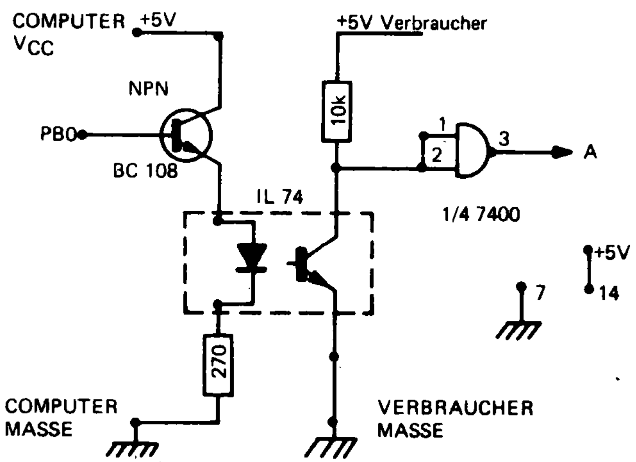
2.124 Schalten eines OPTO-Kopplers

Relais haben den Nachteil, daß sie relativ hohe Steuerleistungen zum Schalten benötigen und daß mechanische Teile bewegt werden. Deshalb sind sie für schnelle Schaltvorgänge nicht geeignet. Ein anderes Bauelement zur galvanischen Trennung von Rechner und Verbraucher ist ein OPTO-Koppler. Hier wird vom Rechner eine Leuchtdiode geschaltet, die einen, im gleichen Gehäuse befindlichen lichtempfindlichen Transistor ansteuert. Die Anschlußbelegung zeigt Abbildung 2.17.



2.17 Anschlußbelegung eines OPTO-Kopplers

Zwischen den Anschlüssen 1 und 2 befindet sich eine Leuchtdiode, zwischen den Anschlüssen 4, 5 und 6 ein lichtempfindlicher Transistor. Eine aufgebaute Schaltung zeigt Abbildung 2.18. Die Leuchtdiode wird über einen Transistor von PB0 angesteuert. Die Spannungsversorgung erfolgt vom Computer. Auf der Verbraucherseite steuert der Transistor den Eingang eines TTL-NAND Gatters. Die Spannungsversorgung dieser Bauelemente erfolgt über ein Netzteil auf der Verbraucherseite.



2.18 Ansteuerung eines OPTO-Kopplers

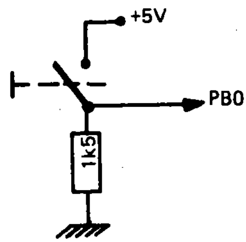
Eine galvanische Trennung sollte immer dann durchgeführt werden, wenn große, vor allem induktive Lasten zu Schalten sind. Diese Trennung ist vor allem dann notwendig, wenn mittels Thyristoren oder Triacs Lasten mit 220 Volt Wechselspannung geschaltet werden.

2.13 Eingabe von Daten über den USER-Port

Werden die Leitungen des USER-Ports als Eingänge programmiert, so können Daten in den Rechner übernommen werden. Das einfachste Beispiel ist die Abfrage einer Taste, ob sie gedrückt ist oder nicht.

2.131 Tastaturabfrage

Die Abbildung 2.19 zeigt den Anschluß einer Taste an den USER-Port. Ist die Taste offen, so liegt die Leitung PB0 über den 1,5k Widerstand an Masse. Ist die Taste geschlossen, so liegen 5 Volt an PB0.



2.19 Abfrage einer Taste

Das Programm in Abbildung 2.20 zeigt die Abfrage einer Taste in den drei Programmiersprachen.

BASIC:

```
100 POKE DB,0
110 A=PEEK(PB)
120 IF A/2=INT(A/2) THEN 110
130 PRINT" TASTE GEDRUECKT"
```

ASSEMBLER:

```
                    AUX      EPZ $02
                    BSOUT    EQU $FFD2

                             ORG $C000
C000: 2026C0                 JSR MAIN
C003: 00                     BRK
```

```
C004: 68       PRINT    PLA
C005: 8502              STA AUX
C007: 68                PLA
C008: 8503              STA AUX+1
C00A: A200              LDX #00
C00C: E602     P1       INC AUX
C00E: D002              BNE *+4
C010: E603              INC AUX+1
C012: A102              LDA (AUX,X)
C014: 297F              AND #$7F
C016: 20D2FF            JSR BSOUT
C019: A200              LDX #0
C01B: A102              LDA (AUX,X)
C01D: 10ED              BPL P1
C01F: A503              LDA AUX+1
C021: 48                PHA
C022: A502              LDA AUX
C024: 48                PHA
C025: 60                RTS

C026: A900     MAIN     LDA #00
C028: 8D03DD            STA DDRB
C02B: AD01DD M          LDA PORTB
C02E: 2901              AND #%00000001
C030: F0F9              BEQ M
C032: 2004C0            JSR PRINT
C035: 544153            ASC \TASTE GEDRUECKT\
C038: 544520
C03B: 474544
C03E: 525545
C041: 434BD4
C044: 60                RTS
```

FORTH:

```
SCR # 19
( EINLESEN                9.11.  EF)
00 DDRB C!
: MES ." TASTE GEDRUECKT" ;
: EIN BEGIN PORTB C@ 1 AND 1 =
  UNTIL MES ;
;S
```

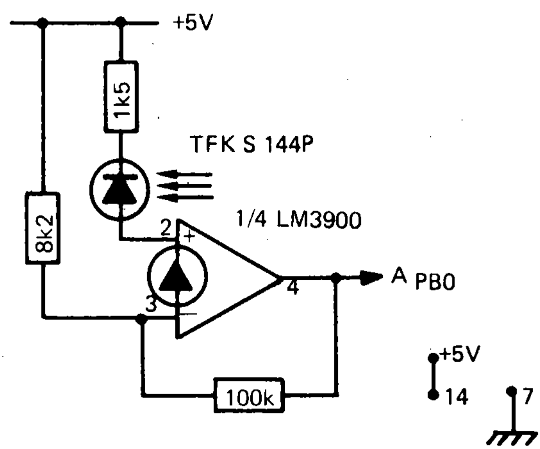
Dabei wird angenommen, daß die Taste über die Leitung PB0 an den Rechner angeschlossen ist und daß bei gedrückter Taste eine Eins eingelesen wird.

In BASIC ist ein Ausblenden mit der UND-Funktion nicht möglich. Deshalb wird in Zeile 120 geprüft, ob eine gerade Zahl anliegt. Dies ist der Fall, wenn die Taste nicht gedrückt ist. In Assembler und FORTH wird die Leitung PB0 mit AND #%00000001, bzw. 1 AND ausgeblendet und das Programm entsprechend verzweigt.

2.132 Lichtdedektor

Die Schaltung in Abbildung 2.21 wird benutzt, um festzustellen, ob eine Lampe brennt oder nicht. Die Eingangsströme der lichtempfindlichen Diode, und der konstante Strom durch einen Widerstand und den Rückkopplungs-Widerstand, werden addiert. Die Dimensionierung ist so ausgelegt, daß die Raumbeleuchtung im Zimmer den Verstärker gerade nicht, eine eingeschaltete Schreibtischlampe ihn aber durchsteuert. Bei Beleuchtung der Diode liegt am Ausgang A eine Eins.

Im Programm wird dieser Ausgang abgefragt und danach entweder "Licht ist ein" oder "Licht ist aus" ausgegeben.



2.21 Lichtdedektor

BASIC:

```
100 POKE DB,0
110 A=PEEK(PB)
120 IF A/2=INT(A/2) THEN 150
130 PRINT "LICHT IST AN"
140 END
150 PRINT "LICHT IST AUS"
160 END
```

ASSEMBLER:

```
                  AUX      EPZ $02
                  BSOUT    EQU $FFD2

                           ORG $C000
C000: 2026C0               JSR MAIN
C003: 00                   BRK

C004: 68          PRINT    PLA
C005: 8502                 STA AUX
C007: 68                   PLA
C008: 8503                 STA AUX+1
C00A: A200                 LDX #00
C00C: E602        P1       INC AUX
C00E: D002                 BNE *+4
C010: E603                 INC AUX+1
C012: A102                 LDA (AUX,X)
C014: 297F                 AND #$7F
C016: 20D2FF               JSR BSOUT
C019: A200                 LDX #0
C01B: A102                 LDA (AUX,X)
C01D: 10ED                 BPL P1
C01F: A503                 LDA AUX+1
C021: 48                   PHA
C022: A502                 LDA AUX
C024: 48                   PHA
C025: 60                   RTS

C026: A900        MAIN     LDA #00
C028: 8D03DD               STA DDRB
C02B: AD01DD               LDA PORTB
C02E: 2901                 AND #%00000001
```

```
C030: F010             BEQ M
C032: 2004C0           JSR PRINT
C035: 4C4943           ASC \LICHT IST AN\
C038: 485420
C03B: 495354
C03E: 2041CE
C041: 60               RTS
C042: 2004C0 M         JSR PRINT
C045: 4C4943           ASC \LICHT IST AUS\
C048: 485420
C04B: 495354
C04E: 204155
C051: D3
C052: 60               RTS
```

FORTH:

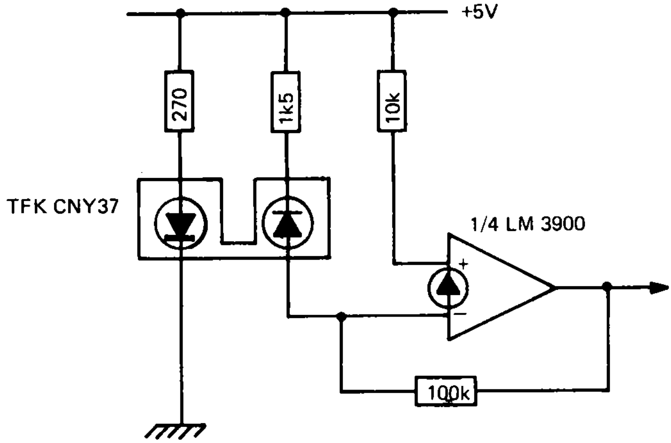
```
SCR # 15
( I/O PHOTODIODE          11.11.EF)
HEX
00 DDRB C! DECIMAL
: MES1 ." LICHT IST AN" ;
: MES2 ." LICHT IST AUS" ;

: E/A  PORTB C@ 1 AND 0=
  IF MES2 ELSE MES1 THEN ;

```

2.22 Programm: Lichtdedektor

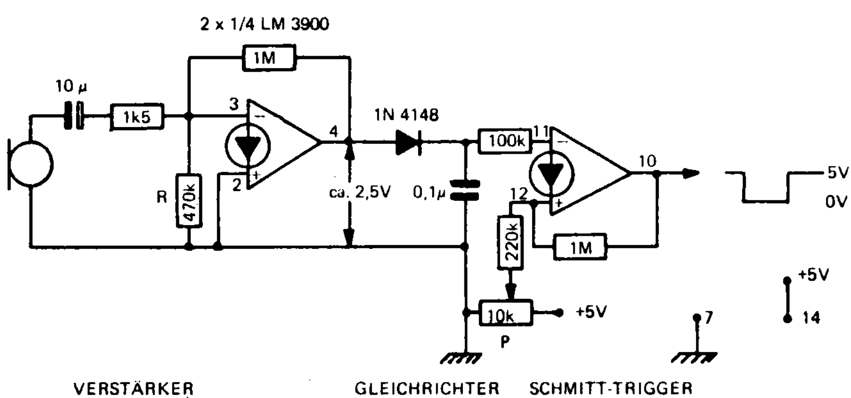
In Abbildung 2.23 ist die Schaltung einer Lichtschranke gezeigt. Als Sensor wird die Gabellichtschranke CNY37 verwendet. In einem Gehäuse ist eine Leuchtdiode untergebracht, die UV-Licht aussendet. Ihr gegenüber liegt eine lichtempfindliche Diode. Wird der Strahlengang unterbrochen, so wird am Ausgang des Verstärkers ein Spannungssprung erzeugt. Diese Lichtschranke soll im nächsten Abschnitt verwendet werden, um die Schwingungsdauer eines Pendels zu messen.



2.23 Lichtschranke

2.133 Akustischer Schalter

In Abbildung 2.24 ist ein akustischer Sensor gezeigt. Im Gegensatz zu lichtempfindlichen Sensoren, treten bei akustischen Sensoren meistens Probleme auf. Diese sind dadurch bedingt, daß die akustischen Aufnehmer, die Mikrofone, meist sehr unterschiedliche Empfindlichkeit und Richtcharakteristik haben. Der allgemeine Aufbau eines akustischen Schalters besteht aus einem Verstärker, einer Gleichrichterstufe und einer Schaltstufe.



2.24 Akustischer Sensor

Die Schaltung in Abbildung 2.14 ist so ausgelegt, daß, ein normal in ein Tonbandmikrofon gesprochenes Wort, ein Signal am Ausgang abgibt. Der Verstärker und die Schaltstufe sind mit dem Operationsverstärker LM3900 aufgebaut, da nur eine Versorgungsspannung von 5 Volt verwendet wird. Die erste Stufe ist als AC-Verstärker geschaltet. Die Verstärkung (hier ca. 700 fach) sollte möglichst hoch sein. Der Widerstand R muß so ausgelegt werden, daß die Gleichspannung am Ausgang ohne Signal ca. 2,5 Volt beträgt. Auf die Gleichrichterschaltung folgt ein Comparator. Durch den Gegenkopplungswiderstand von 1M wird eine Hysterese erzeugt, so daß beim Schalten des Comparators keine Schwingungen auftreten.

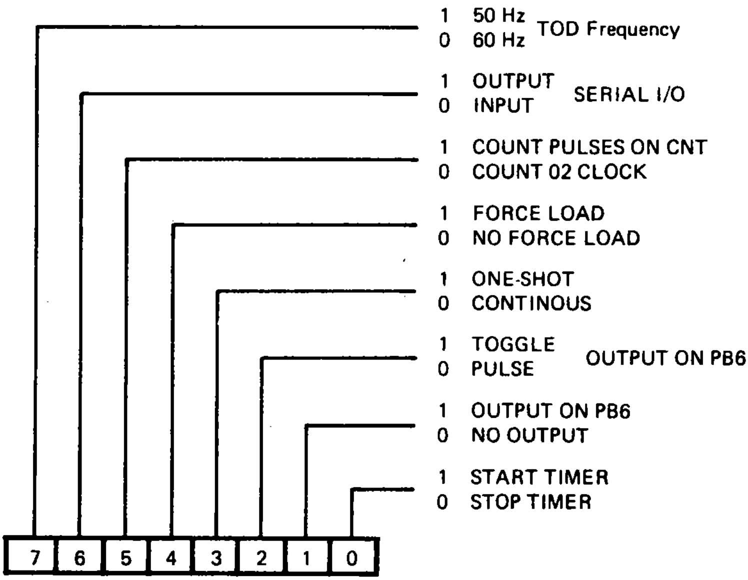
Das Potentiometer P wird so eingestellt, daß die Spannung am Ausgang ohne Signal gleich der Versorgungsspannung ist.

2.2 Programmierung der Timer

Jeder der beiden Timer besteht aus einem 16-Bit Zähler und einem 16-Bit Speicher. Beim Schreiben in den Timer werden die Daten in den Speicher geschrieben. Beim Lesen wird der Zählerstand ausgelesen. Beide Timer können unabhängig voneinander oder, zur Erzeugung von langen Zeitintervallen, miteinander betrieben werden. Die Betriebsart wird durch zwei Steuerregister CRA und CRB bestimmt. Die Bedeutung der einzelnen Bit für die beiden Timer zeigen die Abbildungen 2.25 und 2.26.

Bit CRA0 startet oder stoppt den Timer A. Die Leitung PB6 kann als Impulsausgang verwendet werden. Mit CRA1 = 1 wird PB6 ein Ausgang. Eine Programmierung über das Datenrichtungsregister ist damit unwirksam. An diesem Ausgang kann nun ein Rechteck oder ein Impuls ausgegeben werden. Ist CRA2=1, so wird bei jedem Nulldurchgang des Zählers der Pegel an PB6 geändert. Mit CRA2=0 erscheint an PB6 ein Impuls von der Breite eines Taktimpulses. Bit CRA3 bestimmt, ob der Zähler fortlaufend oder nur einmal arbeiten soll. Ist CRA3=0, so arbeitet der Zähler fortlaufend. Bei jedem Nulldurchgang wird der Inhalt des Speichers neu in den Zähler übernommen. Soll während des Zählens ein neuer Wert sofort übernommen werden, so wird mit CRA4=1 der neue Wert in den Speicher geschrieben. Damit wird der Zähler sofort auf den neuen Zählerstand gesetzt. Ist dagegen CRA4=0,

so wird der neue Wert erst beim nächsten Nulldurchgang übernommen. Bit CRA5 bestimmt, welche Eingangsimpulse zum Zählen verwendet werden sollen. Mit CRA5=0 wird der interne Prozessortakt Ø2 verwendet. Mit CRA5=1 werden Impulse an CNT gezählt.



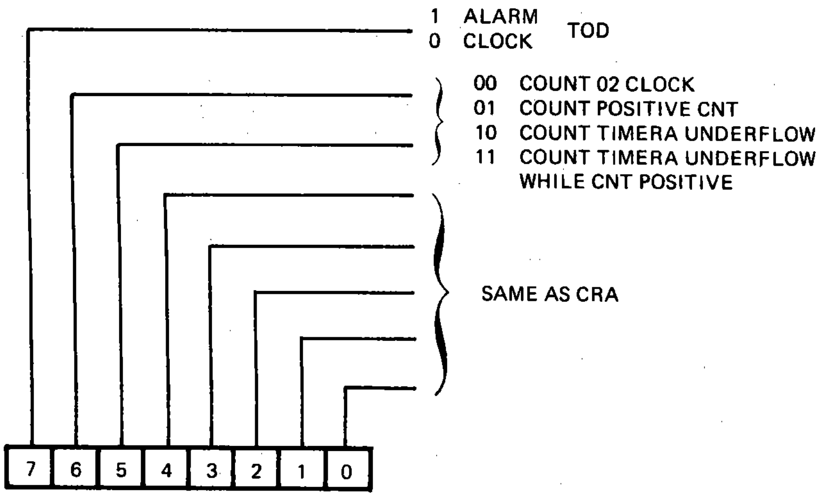
2.25 Steuerregister CRA

Die beiden letzten Bit beeinflussen das Schieberegister und die Tageszeituhr. Mit CRA6=1 können Daten in das Register, mit CRA6=0 können Daten aus dem Register geschoben werden.

CRA7 setzt die Frequenz, mit welcher die Tageszeituhr betrieben wird. Abbildung 2.26 zeigt die Bedeutung der Bit des Steuerregisters CRB. Hier sind nur die Änderungen gegenüber dem Register CRB angegeben.

Bit CRB7 bestimmt, ob die Uhr als Tagesuhr oder als Zeitintervallgeber arbeiten soll.

Die beiden Bit CRB6 und CRB5 legen die Eingangsimpulse für den Timer B fest. Mit CRB6=0 und CRB5=0 wird der interne Ø2 Takt als Eingangsssignal verwendet.



2.26 Steuerregister CRB

Mit CRB6=0 und CRB5=1 werden positive Flanken eines externen Taktgebers am Eingang CNT gezählt. Die beiden Timer können miteinander verbunden werden. Ist CRB6=1 und CRB5=0 so wird im Timer B jeder Nulldurchgang des Timers A gezählt. Bei der letzten Möglichkeit, CRB6=1 und CRB5=1 werden diese Nulldurchgänge nur gezählt, wenn der Eingang CNT positiv ist.

2.21 Rechteckschwingung an PB6

Im ersten Programm in Abbildung 2.27 wird der Timer A freilaufend betrieben, dabei soll sich bei jedem Nulldurchgang der Pegel an PB6 ändern. Für diese Betriebsart muß CRA1=1 und CRA2=1 sein. Ist in die Register T1L und T1H ein Wert übergeben worden, so wird der Timer A mit CRA0=1 gestartet und mit CRA0=0 angehalten. An PB6 erscheint ein Rechtecksignal. Während des Zählens kann ein neuer Wert nach T1 geschrieben und somit die Frequenz geändert werden.

BASIC:

```
100 GOSUB 200
110 POKE CA,7
120 GOSUB 200:GOTO 120
200 INPUT"N=";N
210 IF N<0 THEN END
220 V=INT(N/256)
230 C=INT((N/256-V)*256)
235 POKE L1,C:POKE H1,V
240 RETURN
```

ASSEMBLER:

```
                 AUX     EPZ $02

                         ORG $C000

C000: A502       FREQ    LDA AUX
C002: 8D04DD             STA T1
C005: A503               LDA AUX+1
C007: 8D05DD             STA T1+1
C00A: A907               LDA #07
C00C: 8D0EDD             STA CRA    ;START RECHTECK
C00F: 00                 BRK

C010: A900               LDA #0
C012: 8D0EDD             STA CRA    ;STOP RECHTECK
C015: 00                 BRK
```

FORTH:

```
SCR # 11
  0 ( I/O SQUARE WAVE          9.11.EF)
  1 : FREQ ( N) T1 ! ;
  2 : TON 07 CRA C! ;
  3 : TOFF 00 CRA C! 00 CRB C! ;
```

2.27 Rechtecksignal an PB6

Im BASIC-Programm wird im Unterprogramm ab Zeile 200 eine Zahl eingegeben, die anschließend durch 256 geteilt und in Quotient und Rest aufgeteilt wird. Der Quotient wird in T1H, der Rest in T1L gespeichert. Ist N kleiner Null, so wird das Programm beendet.

Im Assembler-Programm wird angenommen, daß die frequenzbestimmende Zahl in den Zellen $02 und $03 gespeichert ist. Der Inhalt dieser Zellen wird in die Timerregister geschrieben und danach der Timer gestartet.

In FORTH sind folgende Worte verinbart. FREQ (N) speichert N in T1. Da FORTH mit 16-Bit Zahlen arbeitet, kann N direkt in T1L und T1H gespeichert werden. Mit TON wird der Timer gestartet, mit TOFF wieder angehalten.

Ist der Timer einmal gestartet, so läuft er unabhängig von der CPU. Diese kann also andere Aufgaben übernehmen und wird nur benötigt, wenn die Frequenz geändert oder der Timer angehalten wird.

Die Zahl N entspricht der halben Periodendauer. Wird der Timer mit dem internen Takt Ø2 betrieben, so kann die zugehörige Frequenz mit

f=fc/(2*N),

wobei fc die Taktfrequenz des Rechners ist, bestimmt werden Für eine gegebene Frequenz ist N dann:

N=fc/(2*f).

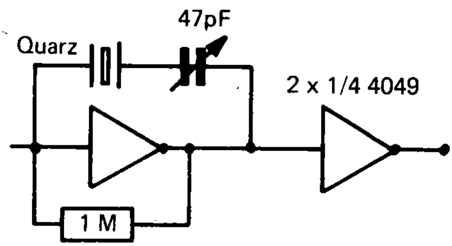
Beim C-64 beträgt die Taktfrequenz ca. 1 MHz. Für eine Rechteckschwingung mit 1kHz berechnet sich N zu 500. Für eine exakte Frequenzangabe sollte fc mit einem Frequenzzähler gemessen werden.

2.22 Impuls- und Periodendauer Messungen

Die Lichtschranke aus Abbildung 2.23 soll nun verwendet werden, um die Geschwindigkeit eines durchlaufenden Gegenstandes und die Periodendauer eines mechanischen Pendels zu messen. Die Messung der Geschwindigkeit führt auf die Messung einer Impulsdauer zurück. Beim

Eintritt des Gegenstandes wird die Lichtschranke unterbrochen, und die Zeit bis zur Freigabe der Lichtschranke wird gemessen.

Um von der Taktfrequenz des Prozessors unabhängig zu sein, wird als Taktgeber ein Quarz mit 1.000000 MHz verwendet. Die Oszillatorschaltung zeigt Abbildung 2.28.



2.28 Oszillatorschaltung mit einem Quarz von 1 MHz

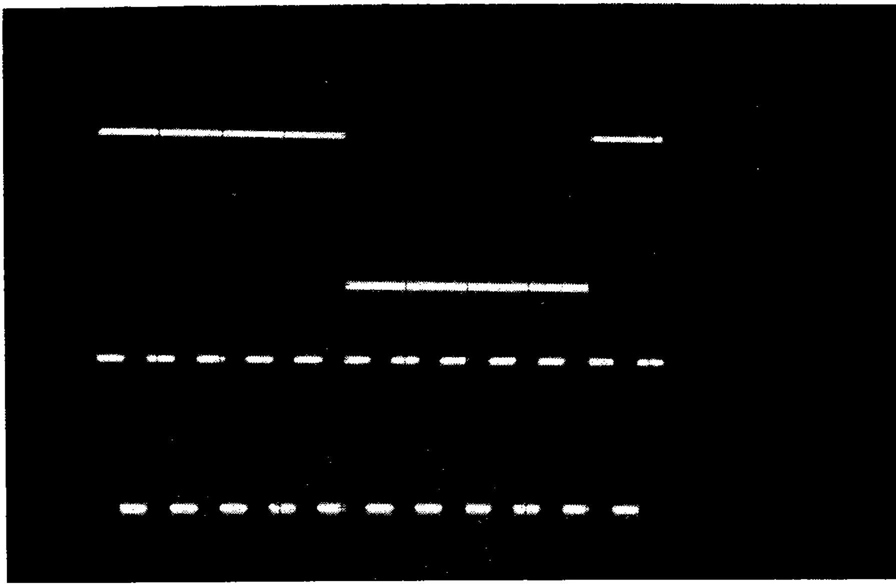
Diese Frequenz wird dem Timer A über CNT2 extern zugeführt. Die Periodendauer des Timers wird so bestimmt, daß an PB6 ein Rechtecksignal von 1 ms auftritt. Der Timer B wird so betrieben, daß bei jedem Nulldurchgang des Timers A, der Timer B um Eins erniedrigt wird.

Bei den Experimenten mit diesen Timern wurde folgendes festgestellt. N ist nicht mehr die halbe Periodendauer, sondern N+1 ist die volle Periodendauer. Die Eingangsfrequenz von 1 MHz für den Timer A wird intern durch den Faktor vier geteilt. Für die Periodendauer von 1 ms ergab sich so der Wert N=249 ((N+1)*4=1000). Leider lag kein Datenblatt für diesen Baustein vor. Ein Hinweis darauf fehlt auch im Programmer's Reference Guide.

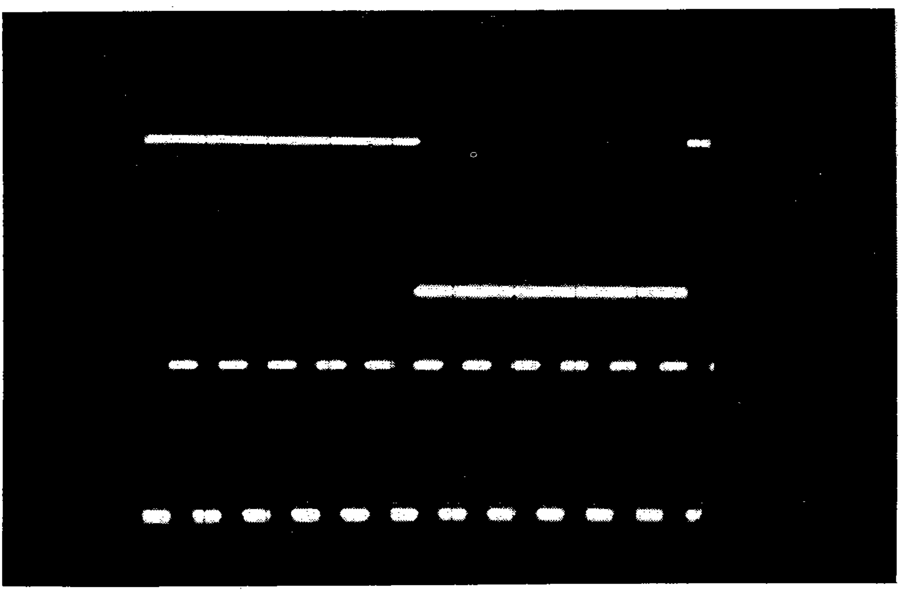
Die beiden folgenden Abbildungen zeigen die Ausgänge der Timer A und B mit dem Teilerverhältniss N=9 (Abbildung 2.29) und N=10 (Abbildung 2.30).

Das Programm für die Impulsdauermessung zeigt Abbildung 2.31.

Die Sprache BASIC ist für solche Messungen zu langsam. Deshalb sollte BASIC nur mit Assembler-Routinen verwendet werden.



2.29 Teilerverhältnis N=9



2.30 Teilerverhältnis N=10

BASIC: zu langsam

ASSEMBLER:

```
                 AUX     EPZ $02

                         ORG $C000

C000: A930     PULSE     LDA #$30
C002: 8502               STA AUX
C004: 8D06DD             STA T2
C007: A975               LDA #$75
C009: 8503               STA AUX+1
C00B: 8D07DD             STA T2+1   ;30000 --> T2,AUX
C00E: A9F9               LDA #249
C010: 8D04DD             STA T1
C013: A900               LDA #0
C015: 8D05DD             STA T1+1   ;1MS --> T1
C018: AD01DD M           LDA PORTB
C01B: 2901               AND #%00000001
C01D: F0F9               BEQ M      ;POSITIVE FLANKE?
C01F: A947               LDA #$47
C021: 8D0EDD             STA CRA
C024: 8D0FDD             STA CRB    ;START TIMER
C027: AD01DD M1          LDA PORTB
C02A: 2901               AND #%00000001
C02C: D0F9               BNE M1     ;NEGATIVE FLANKE?
C02E: A900               LDA #0
C030: 8D0EDD             STA CRA
C033: 8D0FDD             STA CRB    ;TIMER STOP
C036: 38                 SEC
C037: A502               LDA AUX
C039: ED06DD             SBC T2
C03C: 8502               STA AUX
C03E: A503               LDA AUX+1
C040: ED07DD             SBC T2+1
C043: 8503               STA AUX+1 ;ZEIT T BERECHNEN
C045: 00                 BRK
```

FORTH:

```
: F1 ( N) T1 ! ;
: F2 ( N) T2 ! ;
: 1MS 249 F1 ;
: FM 1MS 30000 F2  ;
: CON FM BEGIN PORTB C@ 1 AND
  0=  NOT UNTIL 71 CRA C!
  71 CRB C! ;
: COFF BEGIN PORTB C@ 1 AND
  0= UNTIL TOFF ;
: DT 30000 T2 @ - CR . ." MS" ;
 : PULSE CON COFF DT ;

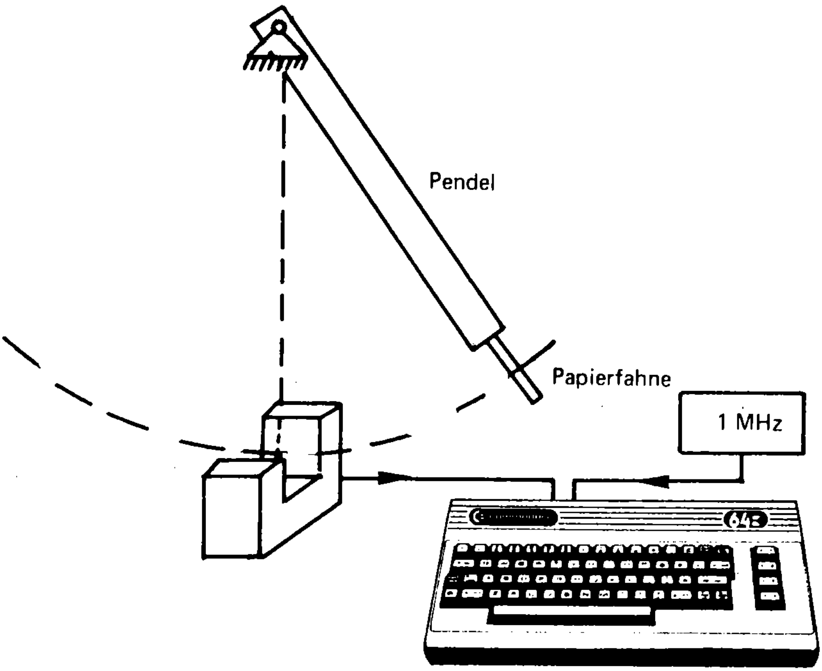
OK
```

2.31 Impulsdauer-Messung

Im Assemblerprogramm wird in den Zellen AUX und AUX+1, sowie im Timer B die Zahl 30000 gespeichert. Dies entspricht einer Gesamtdauer von 30 Sekunden. Im Timer A wird die Zahl 249 gespeichert. Dies entspricht einer Rechteckfrequenz von 1 ms bei einer externen Frequenz von 1 MHz an CNT 2. In der Schleife M wartet das Programm auf eine positive Flanke. Dieses Signal liegt dann an PB0, wenn die Lichtschranke unterbrochen wird. Danach werden die beiden Timer gestartet. Nun wartet das Programm bis die Lichtschranke wieder frei ist. Die beiden Timer werden angehalten und die Zeitdifferenz AUX-T2 berechnet und in AUX gespeichert.

In FORTH wird durch das Wort PULSE eine Messung der Impulsdauer ausgeführt. CON speichert in Timer A die Zahl für 1 ms und in Timer B den Wert 30000. Dann wird solange gewartet, bis der Lichtstrahl unterbrochen wird und die beiden Timer gestartet werden können. Im Wort COFF wird auf die Freigabe der Lichtschranke gewartet. Anschließend werden die Timer angehalten. Das Wort DT berechnet die Differenz zwischen 30000 und dem Inhalt von T2. Dieser Wert wird auf den Bildschirm ausgegeben.

Das Programm zur Periodendauermessung zeigt Abbildung 2.33, die Versuchsanordnung Abbildung 2.32.



2.32 Messung der Periodendauer eines Pendels. Versuchsanordnung

An das drehbar gelagerte Pendel ist am untern Ende eine Papierfahne geklebt. Diese unterbricht beim Hin- und Herschwingen die Lichtschranke. Wird die Lichtschranke zum ersten Mal unterbrochen, so wird die Messung gestartet. Der Impuls beim Zurückschwingen wird übergangen. Bei der nächsten Abdeckung wird die Messung beendet. Im Assemblerprogramm wird im Unterprogramm PFL auf eine positive Flanke (Abdeckung der Lichtschranke) und in NFL auf eine negative Flanke (Freigabe der Lichtschranke) gewartet. Die Warteschleife WAIT wird benötigt, um eine Zeitverzögerung zwischen den Abfragen zu gewährleisten. Dies ist notwendig, da das Umschalten von einem Pegel zum anderen im Analogteil ca. 100 μs beträgt, der Rechner aber wesentlich schneller ist.

In FORTH werden die Worte CON TOFF und DT der Impulsdauermessung verwendet. Das Wort FL wartet auf eine negative, dann auf eine positive Flanke. Eine Messung wird mit dem Wort ?T ausgeführt. Mit CON wird auf die erste positive Flanke gewartet. Nach zwei Pegel-

änderungen ist die Messung beendet. Die Timer werden abgeschaltet und mit DT die Zeitdifferenz berechnet.

BASIC: zu langsam

ASSEMBLER:

```
                    AUX       EPZ $02

                              ORG $C000

C000: 201AC0                  JSR MAIN
C003: 00                      BRK

C004: AD01DD PFL              LDA PORTB
C007: 2901                    AND #%00000001
C009: F0F9                    BEQ PFL
C00B: 60                      RTS

C00C: AD01DD NFL              LDA PORTB
C00F: 2901                    AND #$00000001
C011: D0F9                    BNE NFL
C013: 60                      RTS

C014: A280   WAIT             LDX #$80
C016: CA     W                DEX
C017: D0FD                    BNE W
C019: 60                      RTS

C01A: A930   MAIN             LDA #$30
C01C: 8502                    STA AUX
C01E: 8D06DD                  STA T2
C021: A975                    LDA #$75
C023: 8503                    STA AUX+1
C025: 8D07DD                  STA T2+1   ;30000 --> T2,AUX
C028: A9F9                    LDA #249
C02A: 8D04DD                  STA T1
C02D: A900                    LDA #0
C02F: 8D05DD                  STA T1+1   ;1MS --> T1
C032: 2004C0                  JSR PFL
C035: A947                    LDA #$47
C037: 8D0EDD                  STA CRA
```

```
C03A: 8D0FDD          STA CRB    ;START TIMER
C03D: 2014C0          JSR WAIT
C040: 200CC0          JSR NFL
C043: 2004C0          JSR PFL
C046: 2014C0          JSR WAIT
C049: 200CC0          JSR NFL
C04C: 2014C0          JSR WAIT
C04F: 2004C0          JSR PFL
C052: A900            LDA #0
C054: 8D0EDD          STA CRA
C057: 8D0FDD          STA CRB    ;TIMER STOP
C05A: 38              SEC
C05B: A502            LDA AUX
C05D: ED06DD          SBC T2
C060: 8502            STA AUX
C062: A503            LDA AUX+1
C064: ED07DD          SBC T2+1
C067: 8503            STA AUX+1  ;ZEIT T BERECHNEN
C069: 00              BRK
```

FORTH:

```
SCR # 16
( PERIODENDAUERMESSUNG 15.11.EF)
: FL BEGIN PORTB C@ 1 AND 0=
  UNTIL 10 0 DO LOOP
  BEGIN PORTB C@ 1 AND 1 =
  UNTIL ;

: ?T CON FL FL TOFF DT ;

;S
```

2.33 Messung der Periodendauer

2.3 Programmierung der Echtzeituhr

Der CIA 6526 besitzt eine Echtzeituhr. Vier Register ($DD08 – $DD0B) enthalten 1/10 Sekunden, Sekunden, Minuten und Stunden. Die Anzeige erfolgt als 12-Stundenuhr mit der Angabe AM (0 bis 11 Uhr 59) und PM (12.00 bis 23 Uhr 59). Bit 7 des Stundenregisters ist bei PM gleich Eins. In den Registern sind die Zahlen als BCD (Binary Coded Decimal) Zahlen gespeichert. Dies bereitet bei der Programmierung in Assembler oder in FORTH keine Schwierigkeit. In BASIC dagegen muß zuerst eine Zahlenwandlung durchgeführt werden. Dies soll an folgendem Beispiel gezeigt werden:

In das Minutenregister soll die Zahl 54 geschrieben werden, wobei die 5 in die oberen vier Bit, die 4 in die unteren vier Bit geschrieben wird. Der Inhalt des Minutenregisters ist dann:

01010100 = $54

Wird durch einen POKE-Befehl in BASIC die Zahl 54 in das Register geschrieben, so ist der Inhalt:

00110110 = $36

Die Zahl 54 ist also als Hexadezimalzahl aufzufassen und muß vor dem POKE-Befehl erst in eine Dezimalzahl gewandelt werden.

Bit CRA7 bestimmt, ob die Uhr mit 50 Hertz (CRA7=1) oder mit 60 Hertz (CRA7=0) betrieben wird.

Bit CRB7 legt fest, ob die Uhr als Zeitgeber (CRB7=1) oder als Uhr betrieben wird. Das Programm zeigt Abbildung 2.34.

BASIC:

```
10 A=56576
20 PB=A+1
30 DB=A+3
40 L1=A+4
50 H1=A+5
60 L2=A+6
70 H2=A+7
```

```
80 CA=A+14
90 CB=A+15
100 SS=A+8
110 S= A+9
120 M= A+10
130 H= A+11
200 INPUT "H= (0-23)";HS
205 IF HS>11 THEN HS=HS-12:AM=1
210 V=HS:GOSUB 300
215 IF AM=1 THEN V=V+128
218 POKE H,V
220 INPUT "M= (0-59)";MI
230 V=MI:GOSUB 300:POKE M,V
240 INPUT "S= (0-59)";SE
250 V=SE:GOSUB 300:POKE S,V
260 INPUT"START (J)";A$
270 POKE CA,128:POKE SS,0
280 END
300 V1=INT(V/10): V2=(V/10-V1)*10
310 V=V1*16+V2
320 RETURN
500 HS=PEEK(H):MI=PEEK(M)
510 SE=PEEK(S):S10=PEEK(SS)
520 V=HS
530 IF V>127 THEN PRINT " PM ";:V=V-128:GOTO 550
540 PRINT " AM ";
550 GOSUB 600
560 V=MI:GOSUB 600
570 V=SE:GOSUB 600
580 END
600 V1=INT(V/16): V2=(V/16-V1)*16
610 V=V1*10+V2
620 PRINT"/";V;
630 RETURN
```

ASSEMBLER:

```
PORTB     EQU $DD01
DDRB      EQU $DD03
T1        EQU $DD04
T2        EQU $DD06
SS        EQU $DD08
```

```
                SEK        EQU $DD09
                MIN        EQU $DD0A
                HRS        EQU $DD0B
                CRA        EQU $DD0E
                CRB        EQU $DD0F

                AUX        EPZ $F8

                           ORG $C000

C000: A980      MAIN       LDA #$80
C002: 8D0EDD               STA CRA
C005: A900                 LDA #00
C007: 8D0FDD               STA CRB
C00A: A5F8                 LDA AUX
C00C: 8D0BDD               STA HRS
C00F: A5F9                 LDA AUX+1
C011: 8D0ADD               STA MIN
C014: A5FA                 LDA AUX+2
C016: 8D09DD               STA SEK
C019: A900                 LDA #00
C01B: 8D08DD               STA SS
C01E: 00                   BRK


                           ORG $C100
C100: AD0BDD               LDA HRS
C103: 85F8                 STA AUX
C105: AD0ADD               LDA MIN
C108: 85F9                 STA AUX+1
C10A: AD09DD               LDA SEK
C10D: 85FA                 STA AUX+2
C10F: AD08DD               LDA SS
C112: 00                   BRK
```

FORTH:

```
SCR # 17
  0 ( ECHTZEITUHR              15.11.EF)
  1 HEX DD08 CONSTANT 1/10
  2 DD09 CONSTANT SEC
  3 DD0A CONSTANT MIN
```

```
DD0B CONSTANT STD
0 CONSTANT AM
1 CONSTANT PM DECIMAL
: ./ 47 EMIT ;
: ?TI STD C@ MIN C@ SEC C@
  1/10 C@ ;
: ?TIME ?TI DROP >R >R DUP 127 >
  IF 128 - ." PM " ELSE ." AM "
  THEN HEX . ./ R> . ./ R> .
  DECIMAL  ;
   -->
```

```
SCR # 18
( ECHTZEITUHR                 15.11.EF)
HEX 80 CRA C! DECIMAL
: STI ( HMSZ) >R >R >R >R IF
  128 ELSE 0 THEN R> + STD C!
  R> MIN C! R> SEC C! R> 1/10
  C! DECIMAL ;
```

2.34 Programmierung der Echtzeituhr

Beim Stellen der Uhr werden die Register der Reihe nach, beginnend mit dem Stundenregister beschrieben. Wird ein Wert in das Zehntel-Sekunden Register geschreiben, dann wird die Uhr gestartet.

Wird beim Auslesen der Uhr das Stundenregister gelesen, so wird der Inhalt aller Register in einen Zwischenspeicher übernommen. Dieser Zwischenspeicher kann erst wieder neu beschrieben werden, wenn das Zehntel-Sekunden Register gelesen wurde.

Das BASIC-Programm beginnt in Zeile 200. Hier werden nacheinander die Stunden, Minuten und Sekunden eingegeben. Die Umwandlung von hexadezimal in dezimal erfolgt im Unterprogramm ab Zeile 300. In Zeile 260 wartet das Programm auf den Start der Uhr. Nach Betätigen der RETURN-Taste wird die Uhr gestartet.

Ein Ablesen der Uhr ist mit GOTO 500 möglich. Die Register werden

gelesen und in Hexadezimalzahlen gewandelt. Das Ergebnis wird auf den Bildschirm ausgegeben. Der Sprung nach 500 funktioniert nur dann, wenn in der Zwischenzeit keine Programmänderung durchgeführt wurde. Sonst muß ein getrenntes Programm, mit neuer Variablen-vereinbarung geschrieben werden.

Im Assemblerprogramm wird Zeit für das Setzen der Uhr aus den Zellen F8 bis FA geholt. Dies geschieht im Programm ab Adresse C000.

Beim Auslesen der Uhr wird der Inhalt der Register in den gleichen Zellen gespeichert.

In FORTH holt das Wort ?TIME den Inhalt der Register und gibt die Zeit auf den Bildschirm aus, wie z. B.

PM 1 / 40 / 29 .

Dies entspricht der Zeit 13 Uhr 40 Minuten und 29 Sekunden.

Das Wort STI stellt die Uhr auf eine bestimmte Uhrzeit. Die Eingabe von

HEX AM 10 35 0 0 STI

stellt die Uhrzeit auf 10 Uhr 35 Minuten 0 Sekunden und 0 1/10 Sekunden.

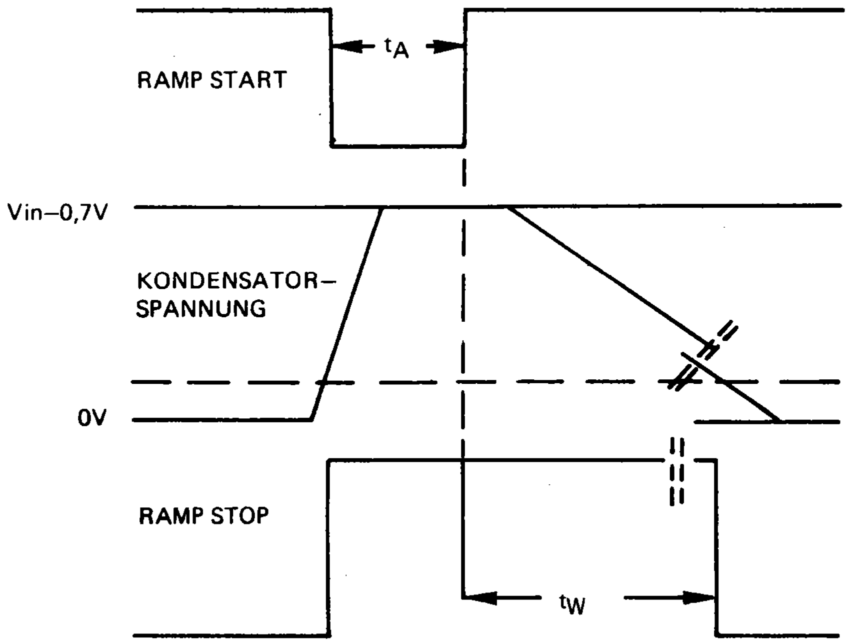
Diese Uhr wird im nächsten Kapitel verwendet werden, um Druck und Temperatur über einen längeren Zeitraum zu verfolgen.

2.4 Anschluß des Analog-Digital Wandlers μA9708 an den USER-Port

Der Analog-Digital Wandler μA9708 ist ein 6-Kanal Wandler mit Decoder, Abtast- und Halteschaltung, Integrator und Referenz-spannungsquelle. Die Funktionsweise des Wandlers zeigt Abbildung 2.35.

Durch eine negative Flanke an RAMP START wird der Kondensator C auf eine Spannung Vin-0.7 Volt aufgeladen. Nach der positiven Flanke

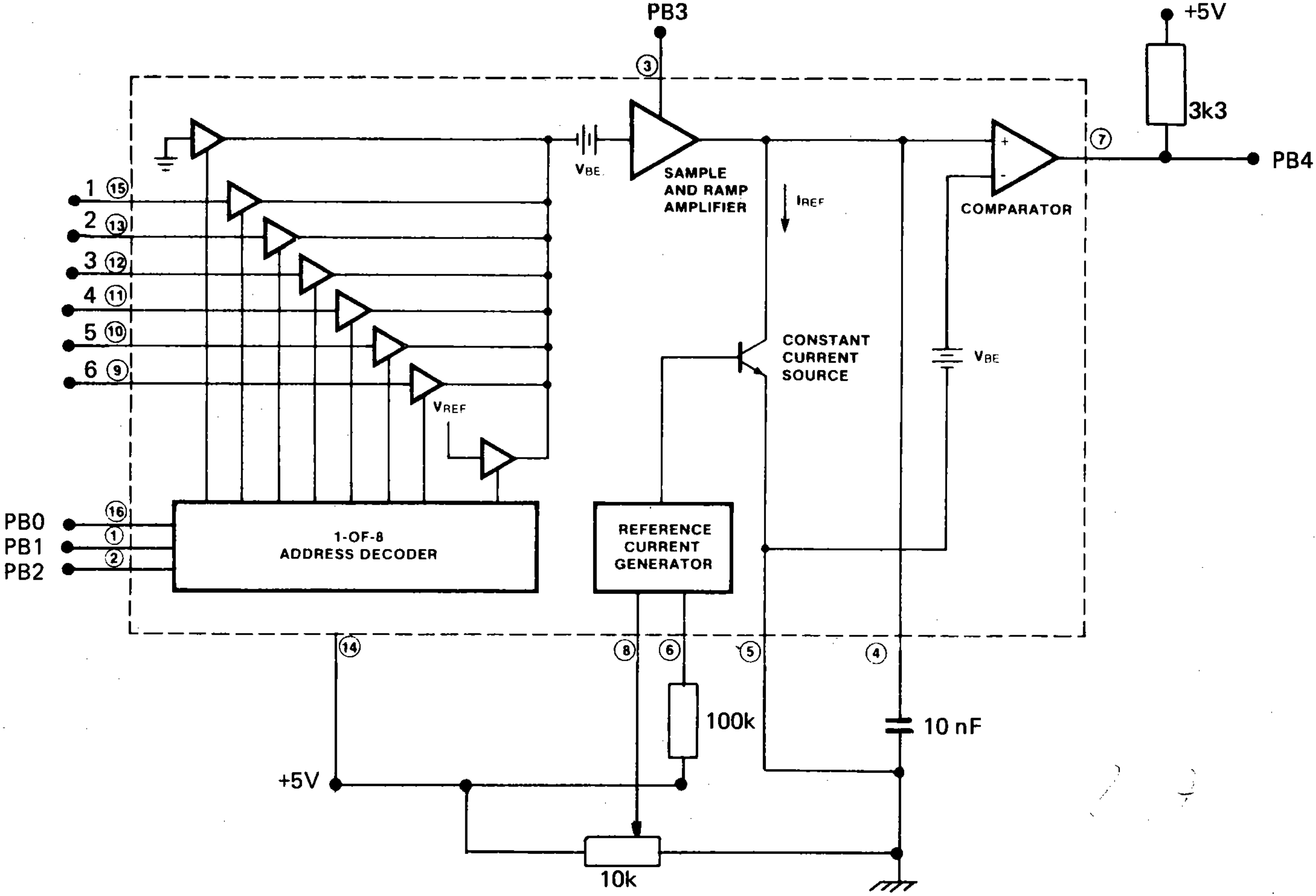
wird der Kondensator mit konstanten Strom entladen. Die Entladezeit wird durch die Referenzspannung bestimmt. Die Spannung am Kondensator wird mit einer konstanten Spannung verglichen. Unterschreitet die Kondensatorspannung diesen Schwellwert, so schaltet ein Komparator zurück. Dies ist die negative Flanke des RAMP STOP Signals. Die Zeit zwischen positiver RAMP START und negativer RAMP STOP Flanke ist ein Mass für die Eingangsspannung.



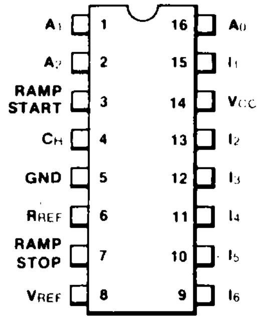
2.35 Funktionsweise des ADW μA9708

Für den Betrieb des Wandlers werden nur 5 Leitungen benötigt. Dies sind die Adressleitungen A0, A1 und A2, die Ausgangsleitung RAMP START und die Eingangsleitung RAMP STOP.

Die Anschlußbelegung zeigt Abbildung 2.36 und die Schaltung 2.37.



2.37 Schaltbild des Wandlers



2.36 Anschlußbelegung

BASIC: nicht implementiert

ASSEMBLER:

```
                                    ORG  $C000

C000: A90F      INIT                LDA  #$0F
C002: 8D03DD                        STA  DDRB
C005: A908                          LDA  #08
C007: 8D01DD                        STA  PORTB
C00A: A9FF                          LDA  #$FF
C00C: 8D04DD                        STA  T1
C00F: 8D05DD                        STA  T1+1
C012: 00                            BRK

                                    ORG  $C020
C020: A9FF      ADW                 LDA  #$FF
C022: 8D04DD                        STA  T1
C025: 8D05DD                        STA  T1+1
C028: A901                          LDA  #01
C02A: 8D01DD                        STA  PORTB
C02D: A230                          LDX  #$30
C02F: CA        AA                  DEX
C030: D0FD                          BNE  AA
C032: A909                          LDA  #09
C034: 8D01DD                        STA  PORTB
C037: A907                          LDA  #07
```

```
C039: 8D0EDD        STA CRA
C03C: AD01DD AB     LDA PORTB
C03F: 2910          AND #%00010000
C041: D0F9          BNE AB
C043: A900          LDA #00
C045: 8D0EDD        STA CRA
C048: 00            BRK
```

FORTH:

```
( ADW UA9708                 21.11.EF)
: INIT 15 DDRB C! 30000 T1 !
  8 PORTB C! ;
: CHA ( N) 8 + PORTB C! ;
( CODE ADW                   22.11.EF)
CODE ADW XSAVE STX, HEX
30 # LDA, T1 STA, 75 # LDA,
T1 1+ STA, DECIMAL
PORTB LDA, 7 # AND, PORTB STA,
48 # LDX, BEGIN, DEX, 0= UNTIL,
8 # ORA, PORTB STA,
7 # LDA, CRA STA,
BEGIN, PORTB LDA, 16 # AND,
0= UNTIL, 0 # LDA, CRA STA,
XSAVE LDX, DEX, DEX, T1 LDA,
BOT STA, T1 1+ LDY, BOT 1+ STY,
NEXT JMP, END-CODE

( ADW TEST                   22.11.EF)
: TT BEGIN INIT 1 CHA ADW . CR
  ?TERMINAL UNTIL ;
0 VARIABLE TMI
: TMIN INIT 0 CHA ADW TMI ! ;
: MESS ( N) CHA ADW TMI @ SWAP
  - . ;
```

2.38 Programm zur Steuerung des Wandlers

Soll der Wandler in BASIC betrieben werden, so müssen die beiden Assemblerroutinen über den SYS Befehl aufgerufen werden.

In dem Programmteil ab Adresse $C000 wird das Tor B des USER-Ports gesetzt und ein Anfangszustand an das Tor ausgegeben. In BASIC ist der BRK-Befehl durch RTS zu ersetzen.

Im Programmteil ab Adresse $C020 wird durch

```
LDA #0X
STA PORTB
```

die Wandlung gestartet. X ist dabei die Kanalnummer 0 bis 7. Nach einer festen Zeitverzögerung zum Aufladen des Kondensators wird durch

```
LDA #0X+8
STA PORTB
```

dieses beendet und der Timer 1 mit

```
LDA #07
STA CRA
```

gestartet. In diesen Timer ist zu Beginn der Wandlung der Wert $FFFF gespeichert worden. Das Tor B wird solange abgefragt, bis das RAMP STOP Signal Null ist. Danach wird der Timer gestoppt. Die Differenz zwischen $FFFF und dem Inhalt der Timerzellen T1 und T1+1 ist die zwischen den beiden Signalen vergangenen Zeit. Für die Ermittlung der Spannung an diesem Kanal muß die Zeitdifferenz zwischen der Wandelzeit von Kanal 0 und dem gemessenen Kanal bestimmt werden.

Im FORTH-Programm wird das Wort INIT definiert. Dieses setzt das Datenrichtungsregister, den Timer und das Tor B. Das Wort CHA erwartet auf dem Stapel die Kanalnummer und speichert sie in Tor B.

Für die eigentliche Wandlung wird auch in FORTH ein Maschinenunterprogramm verwendet. Das Wort ADW ist mit einem FORTH-Assembler geschrieben.

Das Wort TMIN bestimmt die Wandelzeit für Kanal 0 und speichert den Wert in der Variablen TMI. Eine Messung wird mit dem Wort

MESS ausgeführt. Auf dem Stapel wird die Nummer des Kanals übergeben. Die Differenz zwischen TMI und der gemessenen Zeit wird berechnet und auf den Bildschirm ausgegeben.

Eine Messreihe wird mit

TMIN

begonnen. Danach kann mit

3 MESS

die Spannung an Kanal 3 gemessen werden. Die Eingabe von

7 MESS

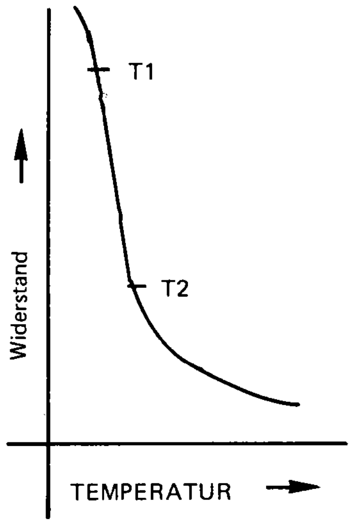
ergibt die Zeitdauer für die größte Eingangsspannung. Aus diesem Wert kann der Skalierungsfaktor bestimmt werden.

Dieser Wandler kann vor allem für Temperaturmessungen verwendet werden. An die einzelnen Kanäle können Temperaturfühler geschaltet werden, die nun die Temperatur an verschiedenen Stellen messen.

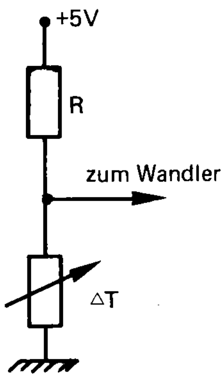
Als Temperaturfühler können zum Beispiel Thermistoren eingesetzt werden. Dies sind temperaturempfindliche Bauelemente. Abbildung 2.39 zeigt die prinzipielle Kennlinie eines Thermistors mit negativen Temperaturkoeffizienten. Diese ist nur im Bereich zwischen T1 und T2 linear.

In dieser Zeichnung sind deshalb keine Zahlenwerte angegeben, da es sehr viele verschiedene Ausführungen gibt. Der Anschluß an den Wandler geschieht am besten als Spannungsteiler nach Abbildung 2.40.

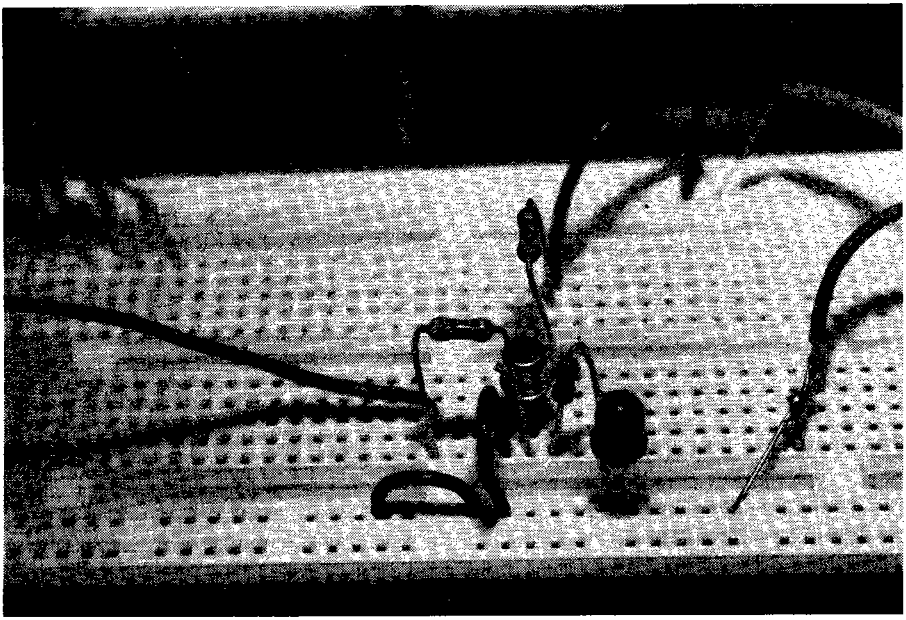
Der Widerstand R muß an den Spannungsbereich des Wandlers angepasst werden. Die Schaltung sollte mit einem Thermometer geeicht werden.



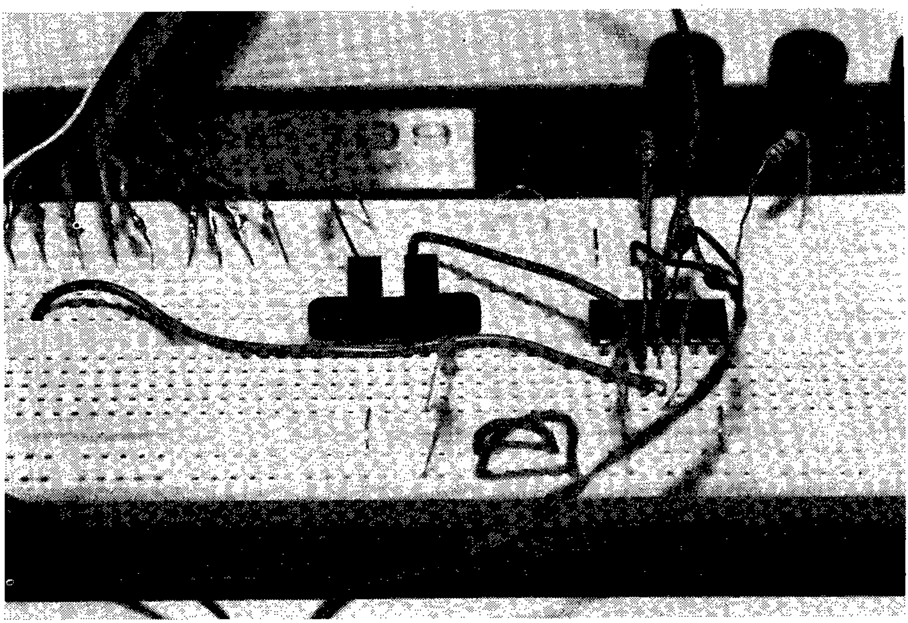
2.39 Kennlinie eines Thermistors



2.40 Anschluß eines Thermistors an den Wandler



Fotodiode

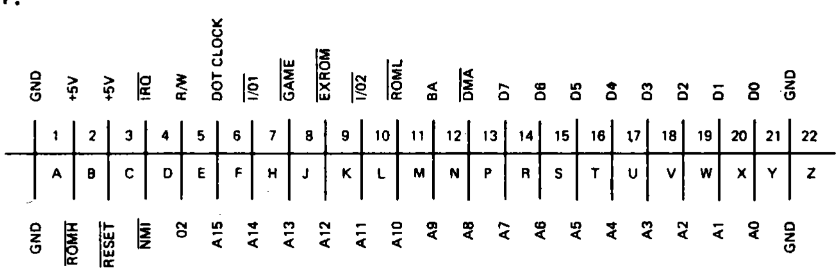


Lichtschranke

3 Hardware Erweiterungen über den Expansion Port

3.0 Hardwareerweiterungen über den Expansion Port

Am Expansion Port des C-64 sind alle Adressleitungen, Datenleitungen und Steuerleitungen verfügbar. Die Steckerbelegung zeigt Abbildung 3.1.

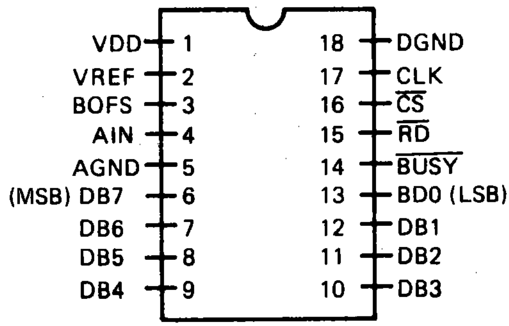


3.1 Steckerbelegung des Expansion Ports

Sollen über diesen Expansion Port externe Bauelemente (Tore, µP kompatible AD/DA-Wandler usw.) angeschlossen werden, so muß im allgemeinen ein bestimmter Adressbereich ausgewählt (decodiert) werden, um dieses Bauelement vom Rechner aus anwählen zu können. Eine derartige Adress-Decodierung wird im nächsten Kapitel gezeigt. In diesem Kapitel soll für die Erweiterungen zwei Adressbereiche verwendet werden, die im C-64 für I/O-Anwendungen vorgesehen sind. Die Auswahlleitung I/01 wird Null, wenn eine Adresse im Bereich DE00 bis DEFF angesprochen wird. Die zweite Leitung I/02 decodiert den Bereich DF00 bis DFFF. Diese 512 Adressen können für Erweiterungen verwendet werden.

3.1 Anschluß des Analog-Digitalwandlers AD7574

Der Analog-Digital Wandler AD7574 der Fa. Analog Device ist ein 8-Bit Converter in CMOS Technik. Er kann direkt an einen Mikroprozessor angeschlossen werden. Die Anschlußbelegung zeigt Abbildung 3.2.

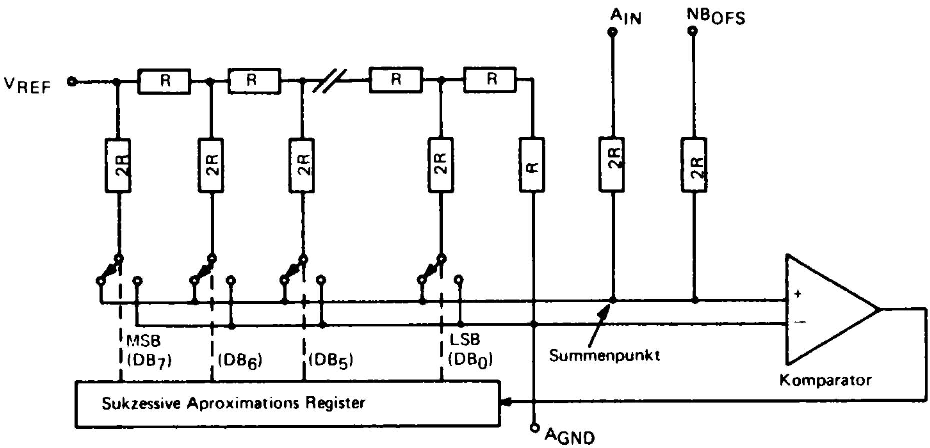


3.2 Anschlußbelegung des ADC 7574

Der ADC wird über den Eingang $\overline{CS}$ angewählt. Mit $\overline{CS}$=L und $\overline{RD}$=H wird die Wandlung einer Spannung in eine Digitalzahl ausgelöst. Diese Umwandlung geschieht nach dem Prinzip der sukzessiven Aproximation. Auf dieses Verfahren wird gleich noch eingegangen. Während des Wandelns ist das Signal $\overline{BUSY}$=L. Durch eine negative Flanke am RD-Eingang werden die Daten an die Ausgänge DB0 bis DB7 gelegt. Die Wandelzeit wird durch ein RC-Glied festgelegt und beträgt 15 bis 20 µs. Der Eingangsspannungsbereich ist von der Spannung VREF abhängig. Mit VREF=–10 Volt beträgt der Eingangsspannungsbereich 0 bis +10 Volt. Den internen Aufbau zeigt Abbildung 3.3.

Ein 2R/R Widerstandsnetzwerk ist über Schalter mit den Eingängen eines Komparators verbunden. Die unbekannte Eingangsspannung wird mit VREF/2 verglichen. Ist die Eingangsspannung größer, so bleibt der Schalter DB7 auf dem Summenpunkt. DB7 ist Eins. Ist die Eingangsspannung kleiner, so wird der Widerstand 2R an AGND (Analog Ground) gelegt. Der nächste Vergleich wird nun mit VREF/2 + VREF/4 wenn DB7=1 ist oder mit VREF/4, wenn DB=0 ist, durchgeführt. Je nachdem die Eingangsspannung größer oder kleiner ist,

bleibt DB6 am Summenpunkt oder wird auf AGND gelegt. Auf diese Weise wird die unbekannte Eingangsspannung nacheinander (sukzessive) angenähert. Ein Beispiel zeigt Abbildung 3.4.



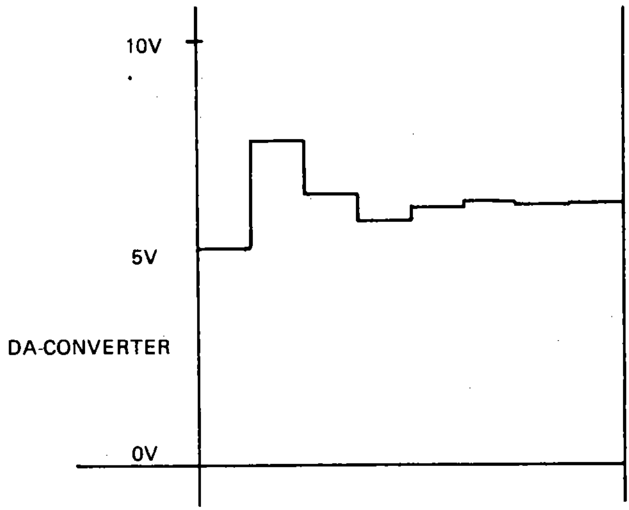
3.3 Interner Aufbau des ADC 7574

Der Analog-Digitalwandler besteht also intern aus einem Widerstands-netzwerk, einem Komparator und einem Taktgeber. Nach dem gleichen Prinzip arbeitet der Analog-Digitalwandler in Kapitel 3.7.
Das Schaltbild zeigt Abbildung 3.5.

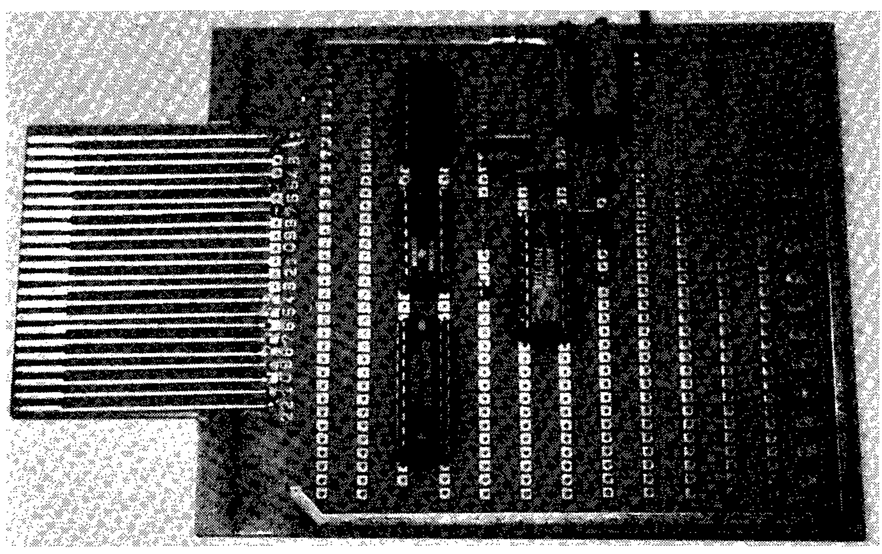
Zwischen den Datenbus des Wandlers und dem Datenbus des C-64 ist ein Bus-Transciever SN74LS245 geschaltet. Diese Trennung des Daten-busses sollte immer dann gemacht werden, wenn mit Experimentier Schaltungen gearbeitet wird. Die Möglichkeit, Bausteine im Inneren des Rechners zu zerstören, ist dann geringer. Beide Bausteine werden durch $\overline{\text{I/O1}}$ angewählt. Die Daten werden von B nach A durchgeschaltet, wenn $\overline{\text{RD}}$=L ist. Dieses Signal wird aus den Signalen $\overline{\text{I/O1}}$ und R/$\overline{\text{W}}$ abgeleitet. Der Wandler benötigt eine Referenzspannung von −10 Volt. Diese kann von einer Zener-Diode oder von der Referenzdiode AD584 geliefert werden.

Beim Aufbau der Schaltung muß auf die Erdung des Analog- und des

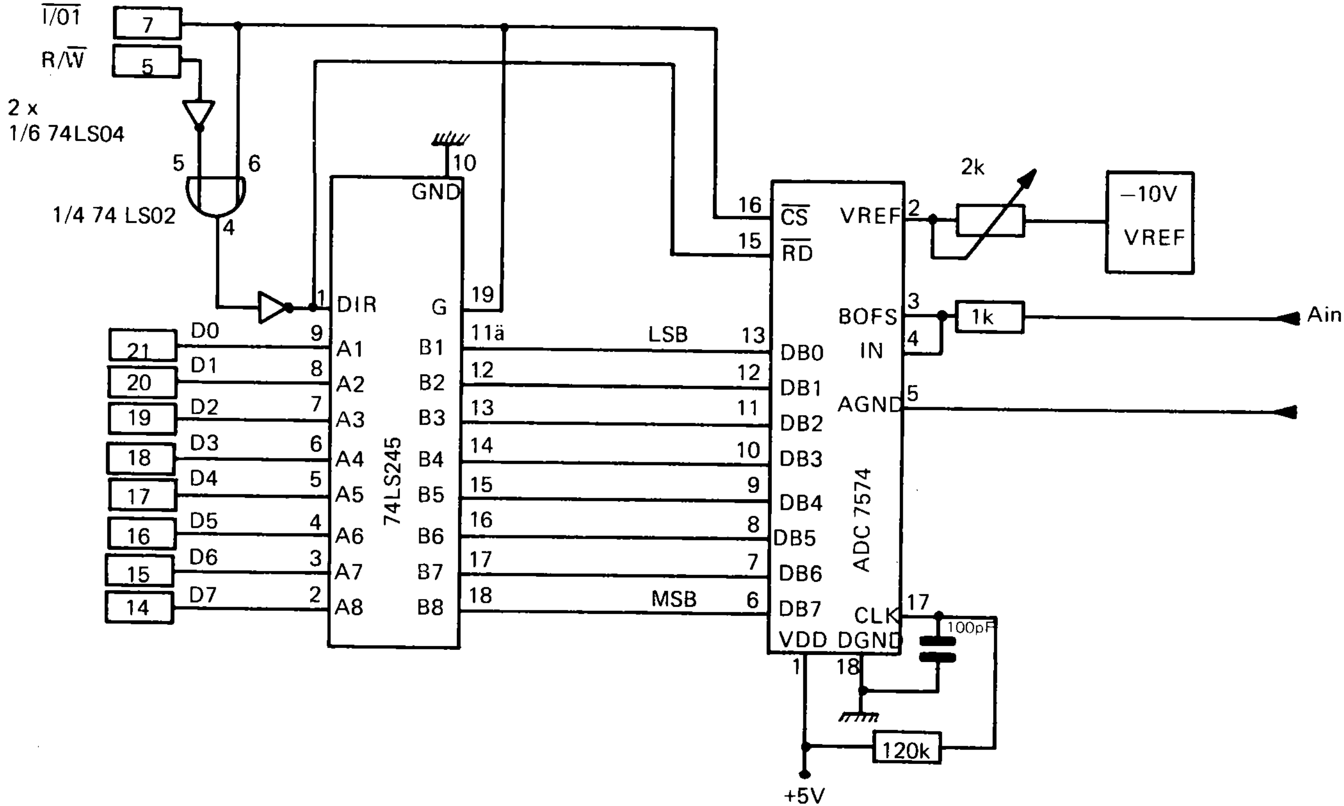
Digitalteils geachtet werden. Alle mit dem Analogteil in Verbindung stehenden Teile müssen mit AGND, alle digitalen Komponenten mit DGND verbunden sein. Fehlmessungen sind sehr oft auf schlechte Masseverbindungen zurückzuführen.



3.4 Analog-Digital Wandlung durch sukzessive Aproximation



A/D-Wandler



3.5 Analog-Digitalwandlung mit dem ADC 7574

Der Wandler wird mit CS=L und RD=H gestartet. Dies entspricht einem WRITE-Befehl an die Adresse $DE00 = 56832. Das folgende Programm wandelt fortlaufend eine Eingangsspannung in einen Zahlenwert um und gibt ihn auf den Bildschirm aus.

BASIC:

```
100 ADW=56832
110 POKE ADW,1:PRINT PEEK(ADW)
120 GOTO 110
```

ASSEMBLER:

```
                      BSOUT    EQU $FFD2
                      ADW      EQU $DE00

                      AUX      EPZ $F8

                               ORG $C000

C000: 2023C0                   JSR MAIN
C003: 00                       BRK

C004: 85F8            PRTBYT   STA AUX
C006: 4A                       LSR
C007: 4A                       LSR
C008: 4A                       LSR
C009: 4A                       LSR
C00A: 2015C0                   JSR PRT
C00D: A5F8                     LDA AUX
C00F: 2015C0                   JSR PRT
C012: A5F8                     LDA AUX
C014: 60                       RTS

C015: 290F            PRT      AND #$0F
C017: C90A                     CMP #$0A
C019: 18                       CLC
C01A: 3002                     BMI P
C01C: 6907                     ADC #$07
C01E: 6930            P        ADC #$30
```

```
C020: 4CD2FF              JMP BSOUT

C023: 8D00DE MAIN         STA ADW
C026: A210                LDX #$10
C028: CA     M            DEX
C029: D0FD                BNE M
C02B: AD00DE              LDA ADW
C02E: 2004C0              JSR PRTBYT
C031: A90D                LDA #$0D
C033: 20D2FF              JSR BSOUT
C036: 4C23C0              JMP MAIN
```

FORTH:

```
HEX
: CONV BEGIN 1 DE00 C! DE00 C@ .
  ?TERMINAL UNTIL ;
DECIMAL
;S
OK
```

3.6 Fortlaufende Wandlung einer Eingangsspannung

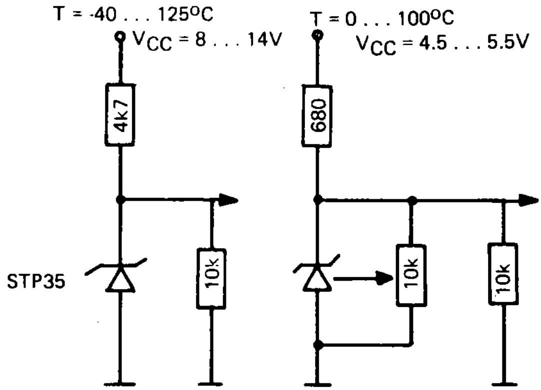
In BASIC und auch in FORTH ist die Zeit zwischen dem Schreib- und Lesebefehl länger als die 20 μs, die der Wandler braucht, um die Spannung in eine Zahl zu wandeln. In Assembler muß dagegen eine Warteschleife programmiert werden. In dieser Schaltung ist es nicht möglich, das BUSY-Signal abzufragen.

Mit folgenden Einstellungen kann der DAC geeicht werden. Die Referenzspannung muß genau –10.000 Volt sein. An den Eingang wird die Spannung 39.1 mV (10.000/256) gelegt. Das Bit DB0 muß zwischen 0 und 1 springen, alle anderen Bit müssen Null sein. Zur Einstellung der Verstärkung wird 9.961 Volt (FS–39 mV) gelegt. Nun wird mit dem Potentiometer P die Verstärkung so eingestellt, das alle Datenbit DB7–DB1 Eins sind, und DB0 zwischen 0 und 1 springt.

Die Grundlagen der Digital-Analog- und der Analog-Digitalwandlung sind im Anhang B zusammengestellt.

3.2 Temperaturmessung mit dem Messfühler STP 35

Der Messfühler STP 35 (Hersteller Texas Instruments) ist ein Präzisions Temperaturfühler. Es stellt intern eine Zener-Diode dar, mit einer Zenerspannung die direkt proportional die absolute Temperatur ist. Bei einer Änderung von 1°K beträgt die Spannungsänderung 10mV. Bei einer Temperatur von +25°C ist die Zenerspannung 2.98 Volt. Diese kann über einen Kalibriereingang eingestellt werden. Abbildung 3.7 zeigt die Schaltung des Temperaturfühlers (nach Unterlagen der Fa. Texas Instruments). Die Messschaltung zeigt Abbildung 3.8.



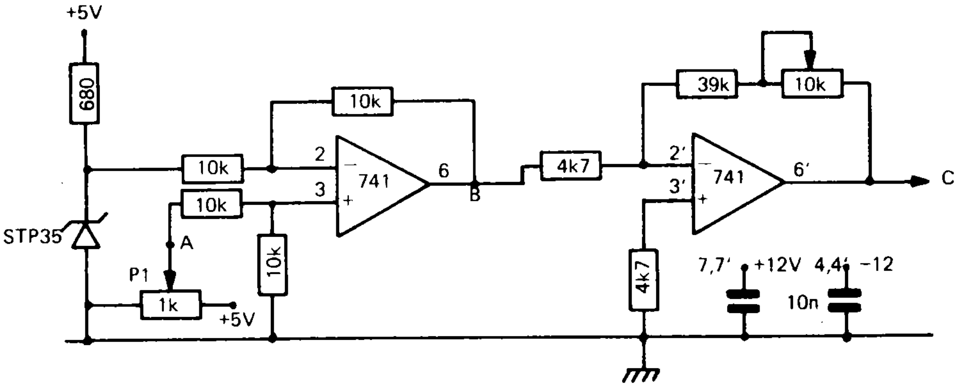
3.7 Schaltung des Temperaturfühlers STP 35

Die Temperatur soll im Bereich 0 bis 100°C gemessen werden. Als Analog-Digitalwandler wird der in 3.1 beschriebene Wandler mit einem Eingangsspannungsbereich von 0 bis 10 Volt eingesetzt werden. Die Ausgangsspannung des Messfühlers muß nun an diesen Bereich angepasst werden.

Bei einer Raumtemperatur von 20°C beträgt die Ausgangsspannung des STP 35 2.93 Volt, bei 0°C 2.73 Volt.

Die erste Stufe der Messschaltung ist ein Differenz-Verstärker mit der Verstärkung 1. Die Eingangsspannung liegt am inventierenden Eingang. Am nichtinvertierenden Eingang liegt eine, über das Potentiometer P1

einstellbare positive Spannung von +5 Volt.



3.8 Temperatur-Messschaltung

Das Potentiometer (10 Wendel Trimmer) wird nun so eingestellt, daß am positiven Eingang die Spannung 2.73 Volt liegt. Die Messung dieser Spannung erfolgt am besten mit einem Digital-Voltmeter. Nun liegt am Ausgang des Differenzverstärkers die Differenz der eingestellten Spannung und der Spannung des Temperaturfühlers. Bei 20°C ist dies die Spannung –0.2 Volt.

Die zweite Stufe der Messschaltung ist ein Verstärker mit einem Verstärkungsfaktor von 10. Dieser wird mit dem Potentiometer P2 (ebenfalls ein 10 Wendeltrimmer) eingestellt. Bei einer Eingangsspannung von –0.2 Volt am Punkt B muß am Ausgang C die Spannung 2.0 Volt liegen. Dies entspricht einer Temperatur von 20°C. Die Empfindlichkeit des Messfühlers ist nach der Verstärkung 100mV/°K. Der Ausgang der zweiten Stufe wird nun an den Eingang des Analog-Digitalwandlers gelegt. Das Programm zeigt Abbildung 3.9.

In BASIC wird in Zeile 110 ein Messwert gewandelt. Der POKE-Befehl startet den Wandler, der PEEK-Befehl liest den Messwert.

Dieser wird ausgegeben. Die Auflösung des Wandlers ist 0.392 Volt. Um die Temperatur zu erhalten, wird A mit diesem Faktor multipliziert. Wenn nun hier ein Wert von z. B. 18.816 (für A=48) ausgegeben wird, so ist es falsch, anzunehmen, daß dies die Temperatur ist. Die Codebreite des 8-Bit Wandlers ist 0.392 Volt (siehe Anhang B). Da immer nur auf ± 1LSB genau gemessen werden kann, so ist dieser Wert, abgesehen von etwaigen Fehlern im Messverstärker, auf ± 0.4 Volt genau. Durch Runden dieser Zahl wird im Programm immer nur der ganz-

zahlige Temperaturwert ausgedruckt.

BASIC:

```
100 ADW=56832
110 POKE ADW,1:A= PEEK(ADW)
120 PRINT A
130 M=A*0.392:PRINT M
140 T=INT(M+0.5):PRINT T
```

ASSEMBLER: nicht implementiert

FORTH:

```
SCR # 22
  0 ( TEMPERATURMESSUNG      23.11.EF)
  1 HEX
  2 : CON ( -N) 1 DE00 C! DE00 C@ ;
  3 DECIMAL
  4 : ROUND SWAP 500 + 1000 / + ;
  5 : SCALE 392 1000 */MOD ROUND ;
  6 : M->T CON SCALE . CR ;
  7
```

3.9 Programm zur Temperaturmessung

Die Sprache FORTH kennt keine Gleitkommazahlen. Diese sind auch bei der Messdatenerfassung nicht notwendig, da die Genauigkeit der Zahlenwerte von der Codebreite der Wandler abhängt.

Zur Umrechnung sind zwei Worte zur Skalierung */ und */MOD vereinbart. Beide erwarten drei Zahlen auf dem Stapel. Mit

48 392 100 */

wird das Produkt 48*392 berechnet. Dieses Zwischenergebnis ist ein 32-bit Produkt. Anschliessend wird durch 100 geteilt. Das Ergebnis ist auf dem Stapel. Beim Wort */MOD wird auch noch der Rest der Division auf dem Stapel abgelegt. Definieren wir ein Wort .SCALE

durch

```
: .SCALE ( N) 392 100 */ . ;
```

so erhalten wir zum Beispiel folgende Umrechnung:

```
48 .SCALE 188
49 .SCALE 192
50 .SCALE 196
```

Die Zahl 192 ist als 19.2 zu interpretieren. Sie besagt aber nur, daß der gemessene Temperaturwert irgendwo zwischen 18.8 und 19.6 Grad Celsius liegt. Im FORTH-Programm wird daher mit */MOD umgerechnet und durch ROUND aufgerundet. Das Wort M–)T führt eine Temperaturmessung durch und gibt den Wert auf den Bildschirm aus.

In FORTH ist es sehr leicht, die Echtzeituhr und die Temperaturmessung zu kombinieren. Sind die Worte für die Echtzeituhr im Wörterbuch vorhanden, so kann durch Einsetzen des Wortes ?TIME vor dem Wort CON die augenblickliche Zeit mit ausgegeben werden.

Verwendet man ein Multitasking FORTH, so kann eine fortlaufende Temperaturmessung über die Echtzeituhr im Hintergrund (Background Task) laufen, während der Rechner im Vordergrund andere Aufgaben erledigt.

Soll in BASIC die Temperaturmessung mit gleichzeitiger Angabe der Uhrzeit durchgeführt werden, so ist in das Ableseprogramm das Wandelprogramm einzubinden.

3.3 Darstellung von Messwerten auf dem Bildschirm

In hochauflösender Grafik können auf dem Bildschirm des C-64 200 Punkte in vertikaler und 320 Punkte in horizontaler Richtung gezeichnet werden. Die Auflösung eines 8-Bit Analog Digitalwandlers ist also höher als die Zahl der Bildpunkte in vertikaler Richtung. Zum Zeichnen der Punkte wird das Programm HIRES-ASSISTENT von H. C. Wagner aus dem Buch (3) verwendet.

```
****************************************
*                                      *
*         HIRES GRAPHIC                *
*                                      *
*         ASSISTENT                    *
*                                      *
****************************************

XCOORD   EPZ $14.5
SECADR   EQU $B9
TEMP     EPZ $FD.E
ADDRESS  EQU TEMP

COLORLOW EQU $0400
COLORHI  EQU $0800

GRAPHICL EQU $2000
GRAPHICH EQU $4000

CHECKCOM EQU $AEFD
GETBYTE  EQU $B79E
GETCOORD EQU $B7EB
GETPARAM EQU $E1D4

BSOUT    EQU $FFD2
LOAD     EQU $FFD5
SAVE     EQU $FFD8

VIDEO    EQU $D000

FALSE    EQU 255
TRUE     EQU 0

CLS      EQU 19+128

         ORG $C000

* INIT HIRES GRAPHIC
* SYS12*4096

C000: 4C18C0          JMP INIT

* CLEAR HIRES SCREEN
* SYS12*4096+3
```

```
C003: 4C33C0            JMP CLEAR

                * SET BACKGROUND COLOR
                * SYS12*4096+6,COLOR

C006: 4C4AC0            JMP COLOR

                * PLOT X,Y (0 <= X < 320)
                *          (0 <= Y < 200)
                * SYS12*4096+9,X,Y

C009: 4C6BC0            JMP SET

                * CLEAR X,Y
                * SYS12*4096+12,X,Y

C00C: 4C67C0            JMP RESET

                * SWITCH HIRES OFF AND
                * BACK TO NORMAL MODE
                * SYS12*4096+15

C00F: 4CD8C0            JMP SWTCHOFF

                * SAVE HIRES GRAPHIC
                * SYS12*4096+18,"NAME",DEVICE

C012: 4CE9C0            JMP SCREENSA

                * LOAD HIRES GRAPHIC
                * SYS12*4096+21,"NAME",DEVICE

C015: 4C00C1            JMP SCREENLO


                * INIT HIRES SCREEN

C018: AD11D0 INIT       LDA VIDEO+17
C01B: 8D52C1            STA SCRATCH+1
C01E: AD18D0            LDA VIDEO+24
C021: 8D51C1            STA SCRATCH
C024: A93B              LDA #27+32
C026: 8D11D0            STA VIDEO+17
```

```
C029: A918              LDA #16+8
C02B: 8D18D0            STA VIDEO+24
C02E: A210              LDX #16
C030: 4C50C0            JMP COLOR1
```

```
                * CLEAR HIRES SCREEN

C033: A000    CLEAR     LDY #0
C035: A920              LDA #GRAPHICL:H
C037: 84FD              STY TEMP
C039: 85FE              STA TEMP+1
C03B: 98      CLEAR1    TYA
C03C: 91FD    CLEAR2    STA (TEMP),Y
C03E: C8                INY
C03F: D0FB              BNE CLEAR2
C041: E6FE              INC TEMP+1
C043: A5FE              LDA TEMP+1
C045: C940              CMP #GRAPHICH:H
C047: D0F2              BNE CLEAR1
C049: 60                RTS
```

```
                * SET BACK COLOR

C04A: 20FDAE  COLOR     JSR CHECKCOM
C04D: 209EB7            JSR GETBYTE

C050: A000    COLOR1    LDY #0
C052: A904              LDA #COLORLOW:H
C054: 84FD              STY TEMP
C056: 85FE              STA TEMP+1
C058: 8A      COLOR2    TXA
C059: 91FD    COLOR3    STA (TEMP),Y
C05B: C8                INY
C05C: D0FB              BNE COLOR3
C05E: E6FE              INC TEMP+1
C060: A5FE              LDA TEMP+1
C062: C908              CMP #COLORHI:H
C064: D0F2              BNE COLOR2
C066: 60      OUTRANGE  RTS
```

* (RE)SET DOT AT X,Y

```
C067: A9FF   RESET    LDA #FALSE
C069: D002            BNE SET1     A.T.
C06B: A900   SET      LDA #TRUE
C06D: 8D53C1 SET1     STA RSFLG
C070: 20FDAE          JSR CHECKCOM
C073: 20EBB7          JSR GETCOORD
C076: E0C8            CPX #200
C078: B0EC            BCS OUTRANGE
C07A: A514            LDA XCOORD
C07C: C940            CMP #320:L
C07E: A515            LDA XCOORD+1
C080: E901            SBC #320:H
C082: B0E2            BCS OUTRANGE
C084: 8A              TXA
C085: 4A              LSR
C086: 4A              LSR
C087: 4A              LSR
C088: 0A              ASL
C089: A8              TAY
C08A: B90FC1          LDA MUL320,Y
C08D: 85FD            STA ADDRESS
C08F: B910C1          LDA MUL320+1,Y
C092: 85FE            STA ADDRESS+1
C094: 8A              TXA
C095: 2907            AND #%00000111
C097: 18              CLC
C098: 65FD            ADC ADDRESS
C09A: 85FD            STA ADDRESS
C09C: A5FE            LDA ADDRESS+1
C09E: 6900            ADC #0
C0A0: 85FE            STA ADDRESS+1
C0A2: A514            LDA XCOORD
C0A4: 2907            AND #%00000111
C0A6: A8              TAY
C0A7: A514            LDA XCOORD
C0A9: 29F8            AND #%11111000
C0AB: 18              CLC
C0AC: 65FD            ADC ADDRESS
C0AE: 85FD            STA ADDRESS
C0B0: A5FE            LDA ADDRESS+1
```

```
C0B2: 6515                    ADC XCOORD+1
C0B4: 85FE                    STA ADDRESS+1
C0B6: A5FD                    LDA ADDRESS
C0B8: 18                      CLC
C0B9: 6900                    ADC #GRAPHICL:L
C0BB: 85FD                    STA ADDRESS
C0BD: A5FE                    LDA ADDRESS+1
C0BF: 6920                    ADC #GRAPHICL:H
C0C1: 85FE                    STA ADDRESS+1
C0C3: A200                    LDX #0
C0C5: A1FD                    LDA (ADDRESS,X)
C0C7: 2C53C1                  BIT RSFLG
C0CA: 1006                    BPL SET2
C0CC: 3949C1                  AND ANDMASK,Y
C0CF: 4CD5C0                  JMP SET3
C0D2: 1941C1 SET2             ORA ORMASK,Y
C0D5: 81FD   SET3             STA (ADDRESS,X)
C0D7: 60                      RTS

             * SWITCH GRAPHIC OFF BACK
             * TO NORMAL MODE

C0D8: AD52C1 SWTCHOFF         LDA SCRATCH+1
C0DB: 8D11D0                  STA VIDEO+17
C0DE: AD51C1                  LDA SCRATCH
C0E1: 8D18D0                  STA VIDEO+24
C0E4: A993                    LDA #CLS
C0E6: 4CD2FF                  JMP BSOUT

             * SAVE HIRES SCREENMEMORY

C0E9: 20FDAE SCREENSA         JSR CHECKCOM
C0EC: 20D4E1                  JSR GETPARAM
C0EF: A200                    LDX #GRAPHICH:L
C0F1: A040                    LDY #GRAPHICH:H
C0F3: A900                    LDA #GRAPHICL:L
C0F5: 85FD                    STA TEMP
C0F7: A920                    LDA #GRAPHICL:H
C0F9: 85FE                    STA TEMP+1
C0FB: A9FD                    LDA #TEMP
C0FD: 4CD8FF                  JMP SAVE
```

```
            * LOAD HIRES SCREENMEMORY

C100: 20FDAE SCREENLO JSR CHECKCOM
C103: 20D4E1          JSR GETPARAM
C106: A961            LDA #6*16+1
C108: 85B9            STA SECADR
C10A: A900            LDA #0
C10C: 4CD5FF          JMP LOAD
             N        EQU 320

            * MULTIPLY TABLE

C10F: 000040 MUL320   DFW 0*N,1*N,2*N,3*N,4*N
C112: 018002
C115: C00300
C118: 05
C119: 400680          DFW 5*N,6*N,7*N,8*N,9*N
C11C: 07C008
C11F: 000A40
C122: 0B
C123: 800CC0          DFW 10*N,11*N,12*N,13*N,14*N
C126: 0D000F
C129: 401080
C12C: 11
C12D: C01200          DFW 15*N,16*N,17*N,18*N,19*N
C130: 144015
C133: 8016C0
C136: 17
C137: 001940          DFW 20*N,21*N,22*N,23*N,24*N
C13A: 1A801B
C13D: C01C00
C140: 1E

            * SET MASK FOR BIT WITHIN
            * THE SELECTED BYTE

C141: 80     ORMASK   DFB %10000000
C142: 40              DFB %01000000
C143: 20              DFB %00100000
C144: 10              DFB %00010000
C145: 08              DFB %00001000
C146: 04              DFB %00000100
```

```
C147: 02              DFB %00000010
C148: 01              DFB %00000001

            * CLEAR MASK FOR BIT WITHIN
            * THE SELECTED BYTE

C149: 7F    ANDMASK   DFB %01111111
C14A: BF              DFB %10111111
C14B: DF              DFB %11011111
C14C: EF              DFB %11101111
C14D: F7              DFB %11110111
C14E: FB              DFB %11111011
C14F: FD              DFB %11111101
C150: FE              DFB %11111110

C151: 0000  SCRATCH   DFW 0  SAVE OLD VALUES
C153: 00    RSFLG     DFB 0  SET OR RESET
C154: 00    YCOORD    DFB 0  YCOORDINATES

PHYSICAL ENDADDRESS: $C155
```

3.10 Program HIRES-ASSISTENT

Mit GRA=12*4096 hat das Programm in BASIC folgende Einsprungadressen:

SYS GRA — Initialisieren der Grafik

SYS GRA+3 — Löschen des Bildspeichers

SYS GRA+6,X — Setzen der Hintergrundfarbe X

SYS GRA+9,X,Y — Zeichnen des Punktes X, Y mit 0<=X<=320 und 0<=Y<=200. Der linke obere Bildpunkt hat den Wert X=0, Y=0.

SYS GRA+15 — Zurück in die normale Bildschirmdarstellung.

SYS GRA+18, "NAME", DEVICE
Speichern des Bildes unter NAME

SYS GRA+21, "NAME", DEVICE
Laden des Bildes in den Bildspeicher. Auf der

Diskette zu diesem Buch ist das Programm unter dem Namen GRA gespeichert. Es kann von dort mit

LOAD"GRA",8,1

geladen werden. Nach dem Laden wird NEW eingegeben. Danach können die BASIC-Programme geladen, bzw. geschrieben werden. Das folgende Programm zeichnet den Spannungsverlauf auf dem Bildschirm.

BASIC:

```
100 ADW=56832
110 GRA=12*4096
200 SYS GRA:SYS GRA+3
300 FOR X=10 TO 300
310 POKE ADW,1:M=PEEK(ADW)
320 P=200-INT(M*200/256)
330 SYS GRA+9,X,P
340 FOR I=1 TO 10:NEXT I
350 NEXT X
360 INPUT A$
370 SYS GRA+15
```

ASSEMBLER: siehe HIRES-ASSISTENT

FORTH:

```
SCR # 23
( MESSWERTE ZEICHNEN   24.11.EF)
( GESCHRIEBEN IN ELCOMP GFORTH )

: WAIT 1000 0 DO LOOP ;
: SCALE ( N-N') 200 256 */
  200 SWAP - ;
: DRAW HGR 311 10 DO CONV SCALE
  I PLOT WAIT LOOP KEY DROP
  TEX ;

( HGR SETZT HIRES GRAFIK          )
( YX PLOT ZEICHNET PUNKT X,Y      )
( TEX SCHALTET IN TEXT MODUS      )


;S
```

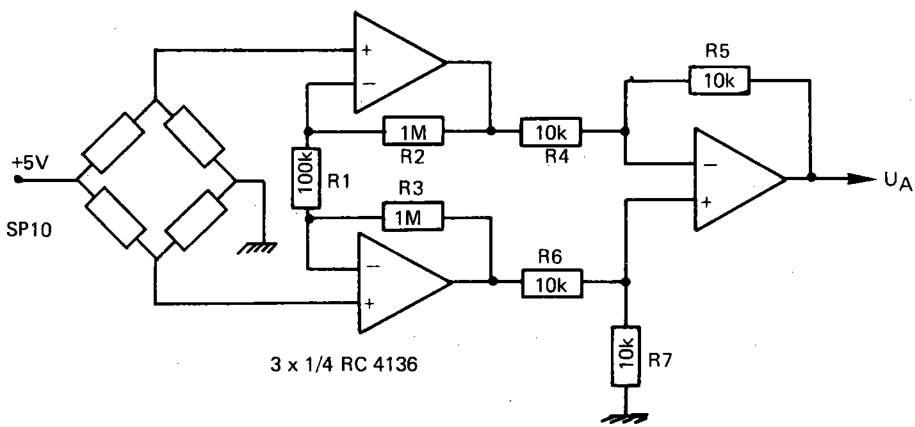
3.4 Druckmessung mit dem SP10

Der SP10 (Hersteller Fa. Texas Instruments) ist ein Druckaufnehmer, der auf einem Silizium-Plättchen aufgebracht ist. In dieses Plättchen ist ein Hohlraum eingeätzt, so daß eine dünne Schicht als Membran übrig belibt. Auf dieser Schicht sind vier Ionen-implantierte Widerstände in Form einer Brückenschaltung aufgebracht. Wird an die Brücke eine Versorgungsspannung gelegt, so ist die Ausgangsspannung proportional zum Druck.

Die Daten des Druckaufnehmers sind:

Ausgangsspannung: 200mV bei einem Druck von 1 Bar.
Empfindlichkeit: 200mV/Bar.

Die Meßschaltung zeigt Abbildung 3.12. Die Brücke ist an einen Instrumentationsverstärker mit der Verstärkung 10 angeschlossen.



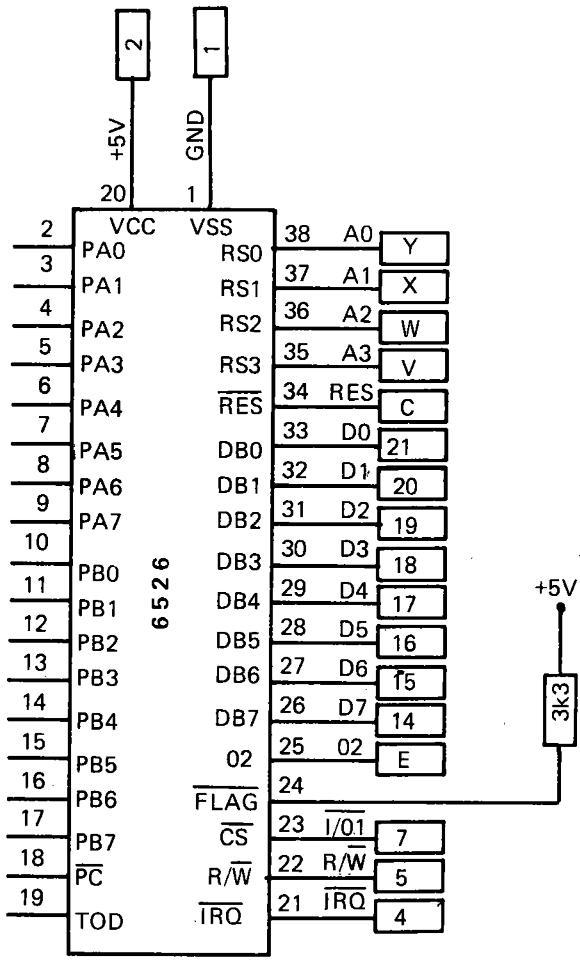
3.12 Druckmessung mit dem SP10

Die Ausgangsspannung muß dem zu messenden Bereich und der Eingangsspannung des Analog-Digitalwandlers angepasst werden. Die Abbildung 3.12 ist auch ein Beispiel, wie Widerstandsbrücken an einen Verstärker angeschlossen werden können.

3.5 Anschluß eines 6526 an den Expansion-Bus

In vielen Fällen reichen die Steuerleitungen am USER-Port nicht aus,

um externe Geräte anzusteuern. An den Expansion Bus kann jedoch sehr leicht ein externer 6526 angeschlossen werden. Dies ist in Abbildung 3.13 gezeigt. Die Auswahl der Register erfolgt über die Adressleitungen A0 bis A3. Wird an das $\overline{CS}$ Signal des 6526 die Leitung $\overline{I/O1}$ gelegt, so haben die Register die Adressen DE00 bis DE0F. Bei Verwendung des $\overline{I/O2}$ Signals, die Adressen DF00 bis DF0F.

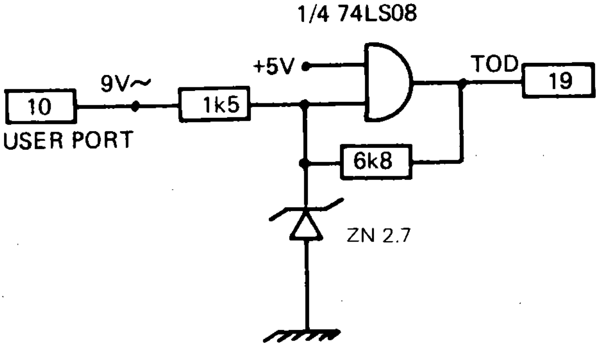


3.13 Anschluß eines 6526 an den Expansion Bus

Der Eingang $\overline{FLAG}$ wird über einen Pull-up Widerstand von 3.3k an die positive Versorgungsspannung gelegt.

Alle anderen Leitungen sind direkt mit dem Expansion Port zu verbinden.

Soll die interne Uhr verwendet werden, so muß an TOD ein Rechtecksignal von 50 Hz gelegt werden. Dieses kann aus der 9 Volt Wechselspannung mit der Schaltung in Abbildung 3.14 erzeugt werden. Die Wechselspannung kann dem Anschluß 10 des USER-Ports entnommen werden (nach Commodore Unterlagen).



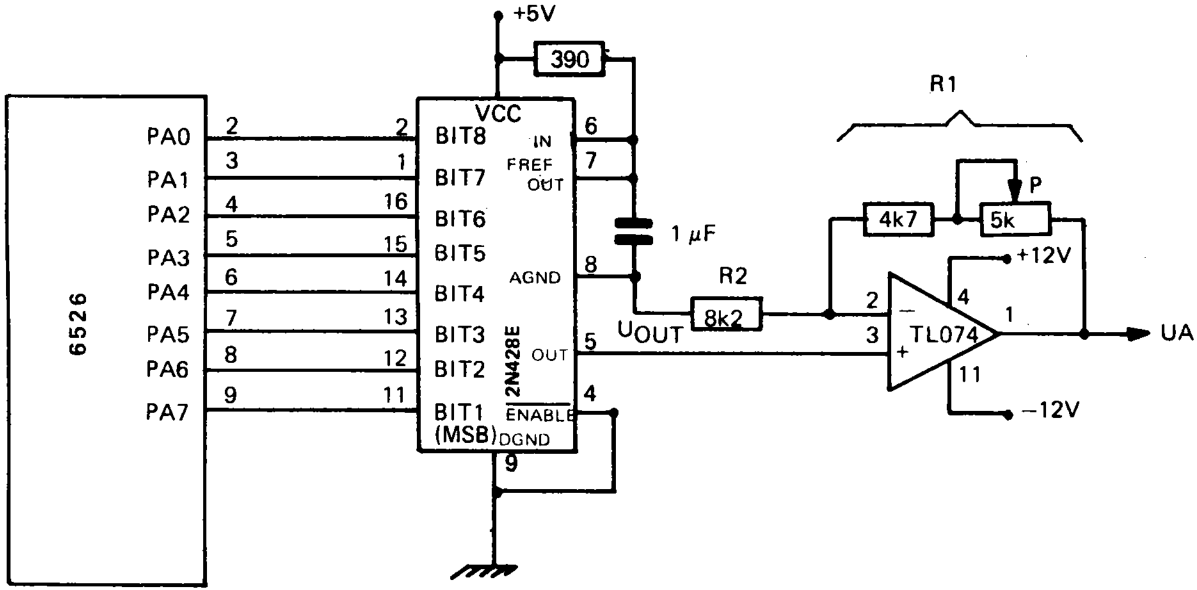
3.14 Erzeugung eines 50Hz Rechtecksignals

Ein Hinweis: Zur Zeit der Drucklegung dieses Buches war es noch nicht gelungen, eine VIA 6522 anzuschliessen. Mehrere Versuche, diesen Baustein zum Funktionieren zu bringen, hatten negativen Erfolg. Die Ursache ist bis jetzt nicht bekannt.

3.6 Anschluß des Digital-Analog-Wandlers ZN428E

Der Digital-Analogwandler ZN428E (Hersteller Fa. FERRANTI) ist ein monolytischer 8-Bit DA Wandler. Die Digitaleingänge sind zwischengespeichert. Somit kann dieser Wandler auch direkt am Datenbus betrieben werden. Mit ENABLE=L sind die Eingänge direkt an die Analogschalter und das R-2R Netzwerk durchgeschaltet. Ist ENABLE=H so entspricht die Ausgangsspannung dem gespeicherten Digitalwert. Änderungen an Digitaleingängen werden nicht auf den Ausgang durchgeschaltet.

In der Schaltung in Abbildung 3.15 ist der Wandler jedoch an einen



3.15 Ansteuerung des Analog-Digital Wandlers ZN428E

6526 angeschlossen. Da nur 8 Leitungen benötigt werden, kann auch Tor B des USER-Ports verwendet werden.

Der Ausgang des Wandlers ist an einen Operationsverstärker angeschlossen. Dieser ist als nicht-invertierender Verstärker beschaltet. Die Ausgangsspannung ist

$$U_A = \left(1 + \frac{R_1}{R_2}\right) U_{OUT} \tag{13}$$

Für Vollausschlag (alle Bit=1) gilt

$$U_A = \left(1 + \frac{R_1}{R_2}\right) U_{REF} \tag{14}$$

Die interne Referenzspannung beträgt 2.5 Volt. Mit der angegebenen Dimensionierung ist der Verstärkungsfaktor 2. Dieser kann mit dem Potentiometer P eingestellt werden. Damit kann der Wandler geeicht werden. Sind alle Bit 1, so muß, bei einem Vollauschlag (FS) von 5 Volt die Spannung 4.82 Volt liegen (siehe Anhang B). In der Schaltung wird als Operationsverstärker 1/4 TL074 verwendet. Dieser kann auch durch einen 741 ersetzt werden.

Zum Eichen kann das folgende Programm benutzt werden:

BASIC:

```
100 PA=56832
110 POKE PA+2,255
130 INPUT"I=";I
135 IF I<0 THEN END
140 POKE PA,I
150 GOTO 130
```

ASSEMBLER: nicht implementiert

FORTH:

```
SCR # 25
  0 ( ADW 428                    24.11.EF)
  1 HEX
  2 FF DE02 C!
  3 : AUS ( N) DE00 C! ;
  4 DECIMAL
  5
```

3.16 Programm zur Eichung des Wandlers

In diesem Programm ist der Wandler an einen externen 6526 mit den Registeradressen DE00 bis DE0F angeschlossen.

Digital-Analogwandler werden in der Praxis zur Erzeugung komplexer Spannungsverläufe eingesetzt. Einige Beispiele zeigen die nächsten Programme.

Das Programm in Abbildung 3.17 erzeugt einen Sägezahn.

BASIC:

```
100 PA=56832
110 POKE PA+2,255
120 FOR I=0 TO 255
130 POKE PA,I
140 NEXT I
150 GOTO 120
```

ASSEMBLER:

```
                    PORTA    EQU $DE00
                    DDRA     EQU $DE02

                             ORG $C000
                    * SAEGEZAHN
C000: A9FF                   LDA #$FF
C002: 8D02DE                 STA DDRA
C005: A200                   LDX #00
C007: 8E00DE M               STX PORTA
```

```
C00A: E8            INX
C00B: D0FA          BNE M
C00D: F0F8          BEQ M
```

FORTH:

```
SCR # 24
  0 ( DAW 428                 24.11.EF)
  1
  2 HEX
  3 FF DE02 C!
  4 : SZ BEGIN 100 0 DO I DE00 C!
  5   LOOP ?TERMINAL UNTIL ;
  6
```

3.17 Sägezahn

Das folgende Programm erzeugt eine Dreieckspannung.

BASIC:

```
100 PA=56832
110 POKE PA+2,255
120 FOR I=0 TO 255
130 POKE PA,I
140 NEXT I
150 FOR I=255 TO 0 STEP -1
160 POKE PA,I
170 NEXT I
180 GOTO 110
```

ASSEMBLER:

```
                               ORG $C000
                     * DREIECK
C000: A9FF                     LDA #$FF
C002: 8D02DE                   STA DDRA
C005: A200                     LDX #00
C007: 8E00DE M                 STX PORTA
C00A: E8                       INX
```

```
C00B: D0FA              BNE M
C00D: A2FF              LDX #$FF
C00F: 8E00DE M1         STX PORTA
C012: CA                DEX
C013: D0FA              BNE M1
C015: F0F0              BEQ M
```

FORTH:

```
: DR BEGIN 100 0 DO I DE00 C!
  LOOP 0 100 DO I DE00 C! -1
  +LOOP ?TERMINAL UNTIL ;
```

3.18 Dreieck

Das nächste Programm erzeugt binäres Rauschen.

BASIC:

```
100 PA=56832
110 POKE PA+2,255
130 I=INT(RND(0)*256)
140 POKE PA,I
150 GOTO 130
```

ASSEMBLER:

```
                PORTA     EQU $DE00
                DDRA      EQU $DE02
                AUX       EPZ $F8

                          ORG $C000
                *RAUSCHEN
C000: A9FF                LDA #$FF
C002: 8D02DE              STA DDRA
```

```
C005: 200EC0 M        JSR RANDOM
C008: 8D00DE          STA PORTA
C00B: 18              CLC
C00C: 90F7            BCC M

C00E: 38     RANDOM   SEC
C00F: 85F9            STA AUX+1
C011: 65FC            ADC AUX+4
C013: 65FD            ADC AUX+5
C015: 85F8            STA AUX
C017: A204            LDX #$04
C019: B5F8   R        LDA AUX,X
C01B: 95F9            STA AUX+1,X
C01D: CA              DEX
C01E: 10F9            BPL R
C020: 60              RTS
```

FORTH:

```
0 VARIABLE RND HERE RND !
: RANDOM RND @ 31241 * 6972 +
  DUP RND ! ;
: RNDNR ( N-N') RANDOM U* SWAP
  DROP ; ( 0 <=N' <N )

HEX FF DE02 C!
: RS BEGIN 100 RNDNR DE00 C!
  ?TERMINAL UNTIL ;
DECIMAL ;S
OK
```

3.19 Rauschen

In BASIC wird die RND-Funktion benutzt, während in Assembler und in FORTH spezielle Zufallszahlengeneratoren verwendet werden.

Komplexe Spannungsverläufe können dadurch erzeugt werden, daß Zahlenwerte aus einer Tabelle geholt und in eine Spannung gewandelt werden.

Während die vorherigen Programme "freilaufend" waren, so können, über die Programmierung der Timer, Spannungsänderungen in gleichen

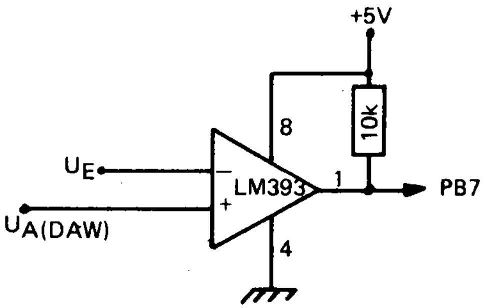
Zeitintervallen erzeugt werden.

Ist ta die Abtastzeit, mit welcher die Zahlenwerte in einer Tabelle in eine Spannung geandelt werden, so ist nach dem SHANNON'schen Abtasttheorem die höchste, noch darstellbare Frequenz gleich 2/ta.

Wird also alle 0.1 ms ein Wert gewandelt, so kann die Ausgangsspannung Frequenzen bis zu 5 kHz enthalten. Höhere Frequenzen, die durch die Unstetigkeiten bedingt sind, müssen durch Tiefpässe abgeschnitten werden.

3.7 Analog-Digital Wandlung mit einem Digital-Analog Wandler

Ein Digital-Analog Wandler kann auch zur Wandlung einer Spannung in einen Zahlenwert eingesetzt werden. Dabei wird wie bei der Hardware Lösung die sukzessive Approximation angewendet. Diese wird durch ein Programm realisiert. Die Schaltung in Abbildung 3.15 wird durch einen Komparator erweitert. In Abbildung 3.20 wird der Komparator LM 393 verwendet. An dem negativen Eingang liegt die zu messende Spannung UE. Im positiven Eingang liegt die Ausgangsspannung des Digital-Analog Wandlers. Der Ausgang des Komparators ist mit dem Eingang PB7 verbunden.



3.20 Erweiterung des DAW mit einem Komparator

BASIC: nicht implementiert

ASSEMBLER:

```
                   PORTA    EQU $DE00
                   PORTB    EQU $DE01
                   DDRA     EQU $DE02
                   BSOUT    EQU $FFD2
                   AUX      EPZ $F8

                   *        ORG $C000

C000: 2046C0                JSR MAIN
C003: 00                    BRK

C004: 85F8         PRTBYT   STA AUX
C006: 4A                    LSR
C007: 4A                    LSR
C008: 4A                    LSR
C009: 4A                    LSR
C00A: 2015C0                JSR PRT
C00D: A5F8                  LDA AUX
C00F: 2015C0                JSR PRT
C012: A5F8                  LDA AUX
C014: 60                    RTS

C015: 290F         PRT      AND #$0F
C017: C90A                  CMP #$0A
C019: 18                    CLC
C01A: 3002                  BMI P
C01C: 6907                  ADC #$07
C01E: 6930         P        ADC #$30
C020: 4CD2FF                JMP BSOUT

C023: A9FF         INIT     LDA #$FF
C025: 8D02DE                STA DDRA
C028: 60                    RTS

C029: A980         CONVERT  LDA #$80
C02B: 85F8                  STA AUX
C02D: A97F                  LDA #$7F
C02F: 8D00DE       CO       STA PORTA
C032: EA                    NOP
C033: EA                    NOP
C034: AC01DE                LDY PORTB
```

```
C037: 3002              BMI C1
C039: 05F8              ORA AUX
C03B: 46F8    C1        LSR AUX
C03D: B004              BCS FIN
C03F: 45F8              EOR AUX
C041: 90EC              BCC C0
C043: 4C04C0 FIN        JMP PRTBYT

C046: 2023C0 MAIN       JSR INIT
C049: 2029C0            JSR CONVERT
C04C: 00                BRK
```

FORTH:

```
SCR # 26
( ADW MIT DAW                    29.11.EF)
HEX
DE00 CONSTANT PORTA
DE02 CONSTANT DDRA
DE01 CONSTANT PORTB
: INIT FF DDRA C! ;
CODE CONV 80 # LDA, N STA,
  7F # LDA,
BEGIN, DROP PORTA STA, NOP, NOP,
  PORTB LDY, 0< NOT IF, N ORA,
  THEN, N LSR, CS NOT IF, N EOR,
  ROT JMP, THEN,
  DEX, DEX, BOT STA, 0 # LDA,
  BOT 1+ STA, NEXT JMP, END-CODE

DECIMAL ;S
OK
```

3.21 Analog-Digital Wandlung mit einem Digital-Analog Wandler

Vom Programm aus wird an den DAW das Bitmuster %01111111 ausgegeben. Im Ausgang des Wandlers liegt dann UFS/2.

In einer Hilfszelle ist das Bitmuster %10000000 gespeichert. Ist die Eingangsspannung UE kleiner als die Ausgangsspannung UA, dann ist der Ausgang des Komparators=H. An PB7 liegt eine Eins. Durch den

Befehl BMI C1 wird der Oder-Befehl übersprungen. Nun wird der Inhalt von AUX eine Stelle nach rechts geschoben. Wandert dabei die Eins in das CARRY-Bit, so ist die Wandlung beendet. Sonst wird durch EOR AUX die höchste Eins im Akkumulator zu Null gemacht. Wenn also die Eingangsspannung UE kleiner als die Spannung UFS/2 ist, dann ist, nach dem ersten Durchlauf des Programms, der Inhalt des Akkumulators gleich

%00111111=$3F ,

der Inhalt der Hilfszelle AUX gleich

%01000000=$40 .

Der Wert $3F wird an den Wandler ausgegeben. Dies entspricht dem Wert UFS/4. Zwischen dem Befehl STA PORTA und dem Befehl LDY PORTB sind zwei NOP Befehle eingefügt. Diese verzögern das Programm, um dem Wandler und dem Komparator Zeit zum Einschwingen zu geben.

Ist nun die Eingangsspannung UE größer als UFS/4, so liegt an PB7 eine Null. Der Befehl BMI C1 wird somit nicht ausgeführt und mit ORA AUX die Eins aus der Zelle AUX in den Akkumulator übernommen. Damit ändern sich die Zelleninhalte wie folgt:

ACCU %00111111

AUX %01000000 .

Nach ORA AUX, LSR AUX und EOR AUX:

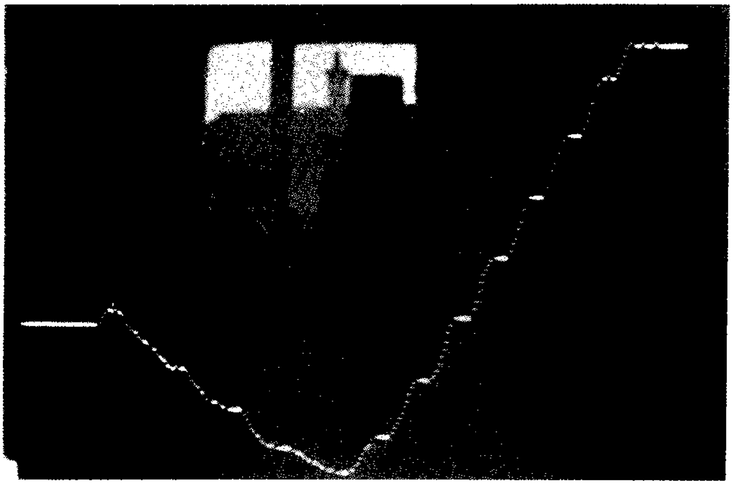
ACCU %01011111

AUX %00100000

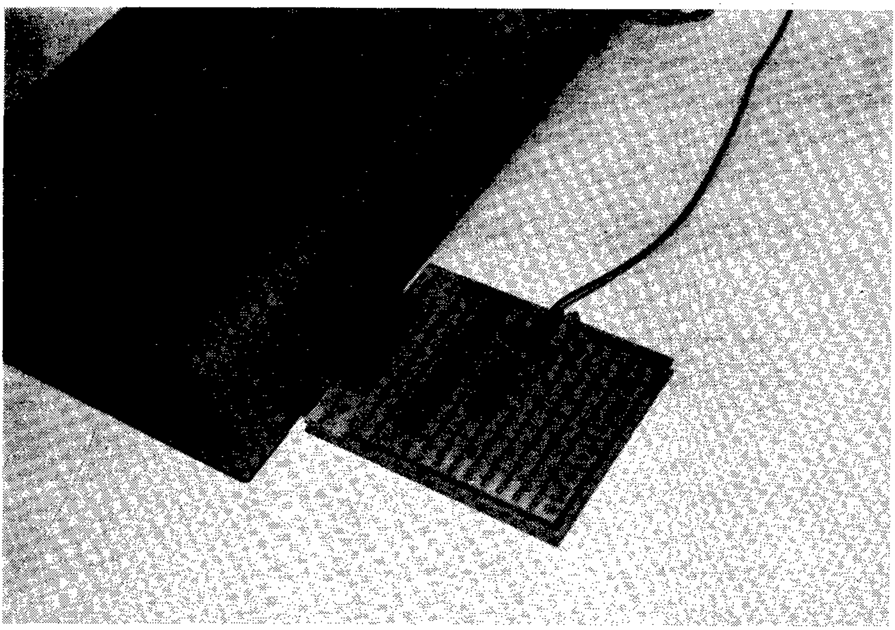
Ist also PB7=0 so bleibt die Eins erhalten, ist PB7=1, so wird das entsprechende Bit Null.

Die Wandelzeit hängt also im wesentlichen von der Programmlaufzeit

ab. Deshalb wird auch der Zustand des Wandlers über PB7 abgefragt. Somit kann mit BMI verzweigt werden. Hätte man ein anderes Bit verwendet, so hätte ein weiterer Befehl AND #%XXXXXXXX eingeführt werden müssen, um das Bit ausblenden zu können.



Darstellung eines Spannungsverlaufes auf dem Bildschirm

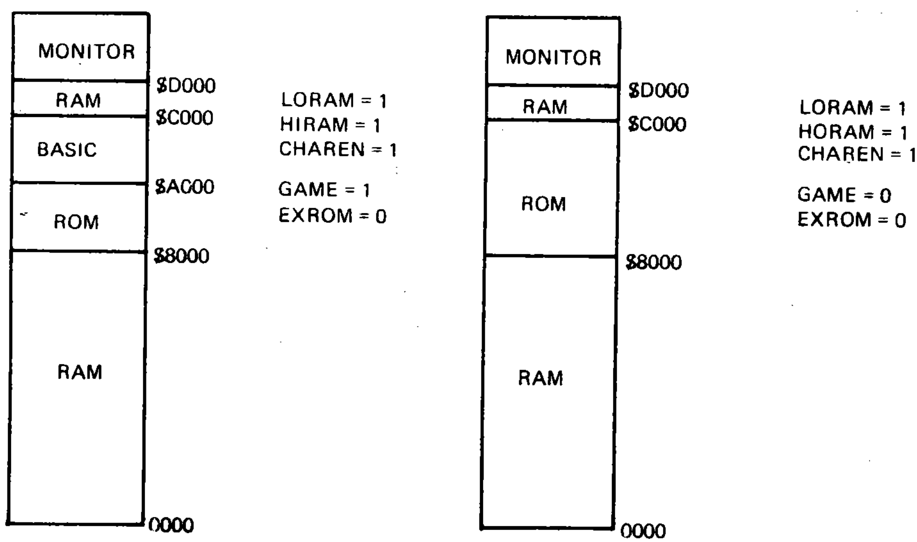


A/D-Wandler in den C-64 eingesteckt

4 Verwendung des ROM-Bereichs für Erweiterungen

4.0 Verwendung des ROM-Bereichs für Erweiterungen

Der C-64 hat einen voll ausgebauten Speicher. Alle 64k sind mit RAM belegt. Durch extern zugeführte Signale können Teile des Speichers abgeschaltet werden. Dafür stehen am Expansion Bus zwei Signale, EXROM und GAME zur Verfügung. Abbildung 4.1 zeigt die möglichen Speicherkonfigurationen.

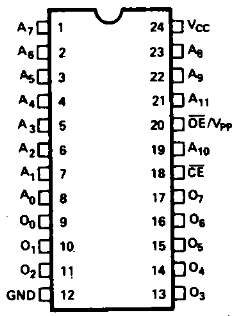


4.1 Speicherverteilung

Im normalen Betrieb ist nach dem Einschalten RAM bis zur Adresse $A000 vorhanden. Wird vor dem Einschalten das Signal EXROM auf Null und GAME auf Eins gelegt, so reicht der für BASIC zur Verfügung stehende RAM-Bereich nur bis zur Adresse $8000. Der Speicherbereich zwischen $8000 und $A000 ist abgeschaltet. Wird auch vor dem Einschalten auch GAME auf Null gesetzt, dann reicht der ROM-Bereich bis zur Adresse $C000. Der BASIC Interpreter ist abgeschaltet. Im ersten Fall kann die Sprache BASIC mit einer zusätzlichen Programm-Kassette verwendet werden.

4.1 Anschluß eines EPROMS 2732

Das EPROM 2732 ist ein 4k*8 Bit Speicher, dessen Inhalt über einen EPROM-Burner programmiert wird. Dieser Inhalt kann durch eine UV-Lichtquelle wieder gelöscht werden. Die Anschlußbelegung zeigt Abbildung 4.2.

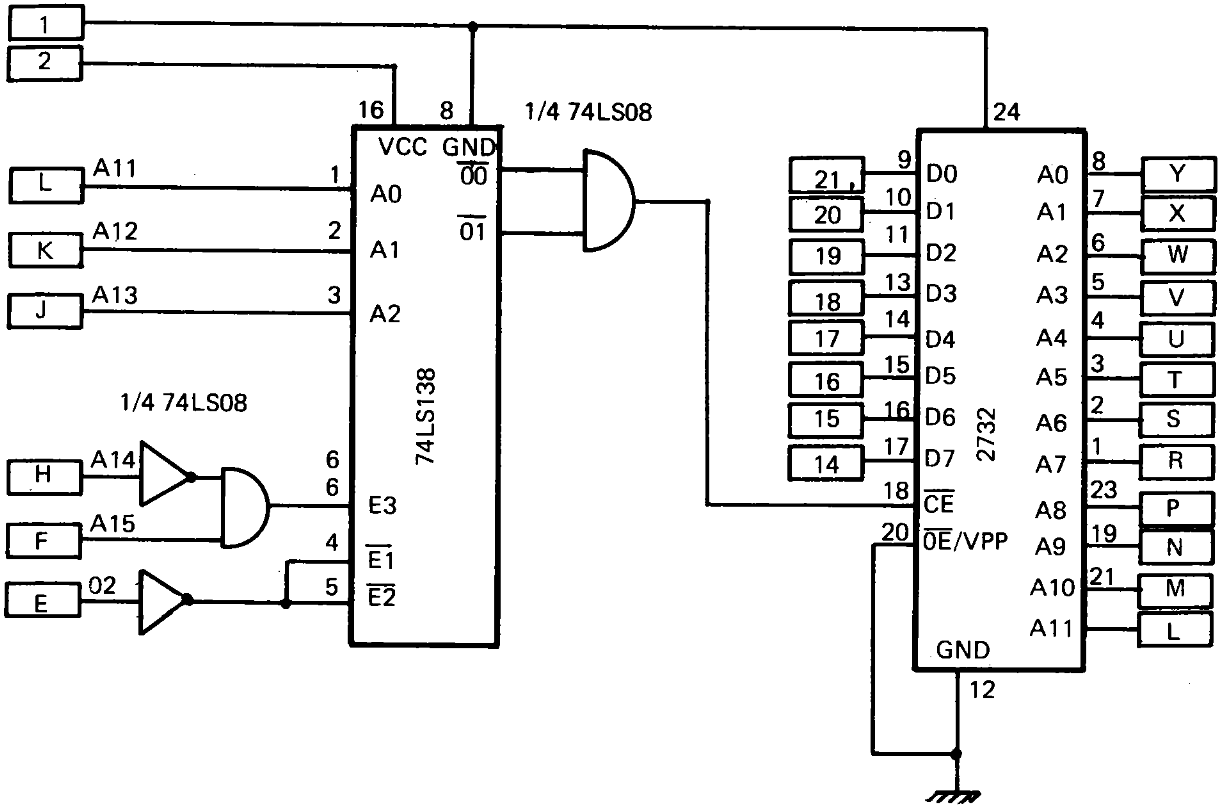


4.2 Anschlußbelegung des EPROMS 2732

Beim Lesen des Inhalts ist $\overline{OE}$/Vpp=L. Die Auswahl des EPROMS erfolgt über die $\overline{CS}$ Leitung. Dieses Signal muß nun aus den Adressleitungen dekodiert werden. Eine Möglichkeit zeigt Abbildung 4.3.

Zur Dekodierung wird ein 1 aus 8 Dekoder/Multiplexer 74LS138 verwendet.

Zum besseren Verständnis ist in Abbildung 4.4 die Wahrheitstabelle dargestellt.



4.3 Adressdekodierung und Anschluß des EPROMS

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

4.4 Wahrheitstabelle des 74LS138

Die Adressleitungen A15 und A14 sind über die Funktion A14 V A15 miteinander verknüpft. Ist A15=H und A14=L dann ist E3=H. Die weitere Dekodierung erfolgt über die Adressleitungen A13, A12 und A11. Der Ausgang O0 ist somit L, wenn eine Adresse zwischen

A15	A14	A13	A12	A11	
H	L	L	L	L	und

A15	A14	A13	A12	A11
H	L	H	H	H

angesprochen wird und zugleich der Ø2 Takt positiv ist. Dies ist der Adressbereich $8000 bis $87FF. Für das 4k EPROM wird auch noch der Adressbereich von $8B00 und $8FFF benötigt. Deshalb werden die beiden Leitungen O0 und O1 über eine UND-Schaltung zusammengefasst und an den $\overline{CE}$ Eingang des EPROMS gelegt. Der Inhalt des EPROMS kann nun im Adressbereich $8000 bis $8FFF gelesen werden.

Für ein zweites EPROM können die Leitungen Ø2 und Ø3 benutzt werden.

4.2 Dekodierung für weitere I/O Bausteine

Abbildung 4.5 zeigt eine Weiterführung der Dekodierung zum Anschluß weiterer I/O Bausteine. Dies ist die gleiche Art der Dekodierung wie sie für die Einschübe des APPLE II benutzt wird. Für den C-64 wurde der Adressbereich von $8000 bis $8FFF ausgewählt, während er beim APPLE II im Bereich $C000 bis $CFFF liegt.

Der Ausgang O1/1 entspricht dem I/O STROBE, die Ausgänge O1/2 bis O7/2 den DEVICE SELECT Signalen und O0/3 bis O7/3 den I/O SELECT Signalen beim APPLE II.

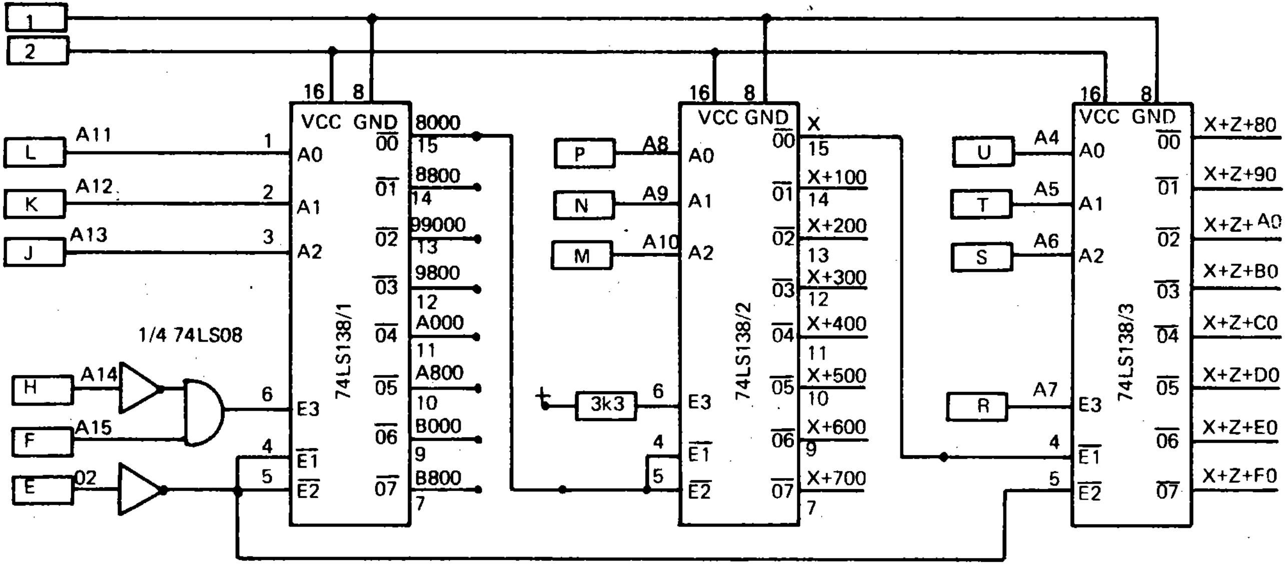
In der Tabelle in Abbildung 4.6 sind die decodierten Adressen nochmals zusammengestellt.

Die in Abbildung 4.7 gezeigte Platine wurde für 6502 (6510) Rechner entwickelt. Sie enthält die Dekodierung nach Abbildung 4.5 und vier Steckplätze. Die Anschlußbelegung dieser Steckplätze entspricht der Belegung, wie sie im APPLE II an den Steckplätzen vorhanden ist. Allerdings sind nicht alle Leitungen des APPLE II angeschlossen.

Die Anschlußbelegung zeigt Abbildung 4.8.

Vom Rechner sind die Adress-Daten und einige Steuerleitungen angeschlossen. Die drei Signale DEVICE SELECT, I/O SELECT und I/O STROBE werden von der Dekodierung erzeugt. Die Spannungen werden von einem Netzgerät zugeführt. Die Steckplätze sind von 1 bis 4 numeriert. Folgende Adressen sind angeschlossen:

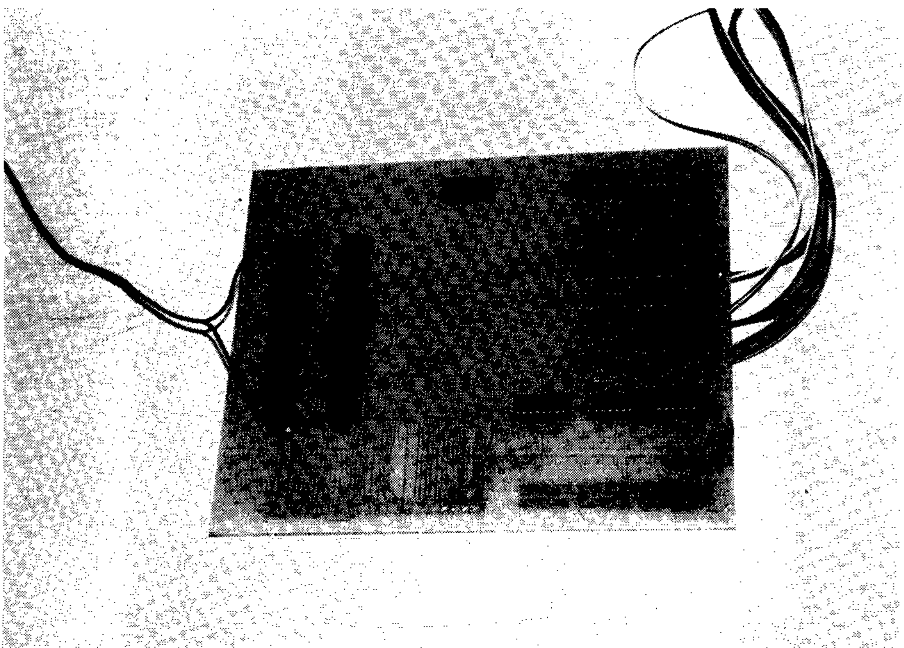
Platz	I/O SELECT	DEVICE SEL	I/O STROBE
1	8100–81FF	8090–809F	8800–8FFF
2	8200–82FF	80A0–80AF	8800–8FFF
3	8300–83FF	80B0–80BF	8800–8FFF
4	8400–84FF	80C0–80CF	8800–8FFF



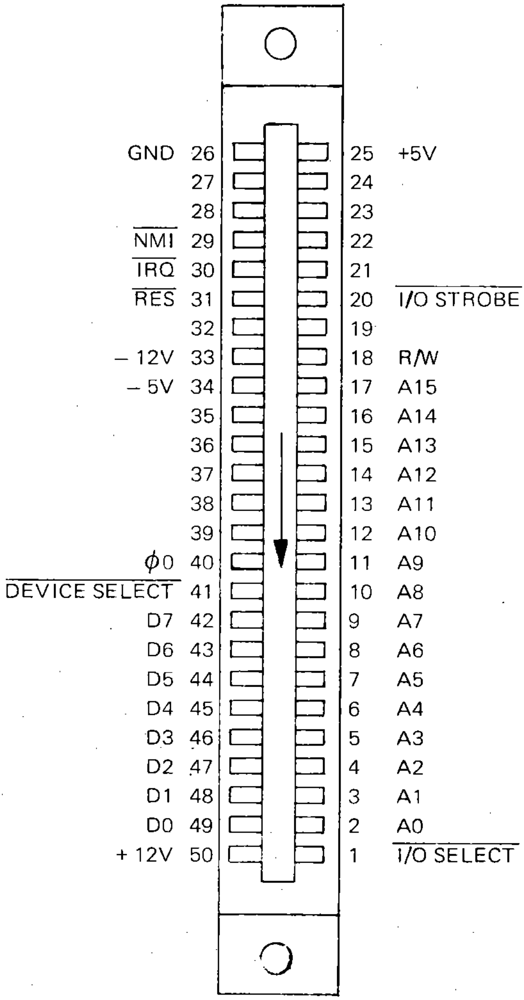
4.5 Dekodierung von Adressen für I/O Bausteine

A15	A14	A13	A12	A11	X	A10	A9	A8	Z	A7	A6	A5	A4	
H	L	L	L	L	8000	L	L	L	X	H	L	L	L	X+Z+80
H	L	L	L	H	8800	L	L	H	X+100	H	L	L	H	X+Z+90
H	L	L	H	L	9000	L	H	L	X+200	H	L	H	L	X+Z+A0
H	L	L	H	H	9800	L	H	H	X+300	H	L	H	H	X+Z+B0
H	L	H	L	L	A000	H	L	L	X+400	H	H	L	L	X+Z+C0
H	L	H	L	H	A800	H	L	H	X+500	H	H	L	H	X+Z+D0
H	L	H	H	L	B800	H	H	L	X+600	H	H	H	L	X+Z+50
H	L	H	H	H	B800	H	H	H	X+700	H	H	H	H	X+Z+F0

4.6 Zusammenstellung der Adressen



4.7 Ein-/Ausgabeplatine

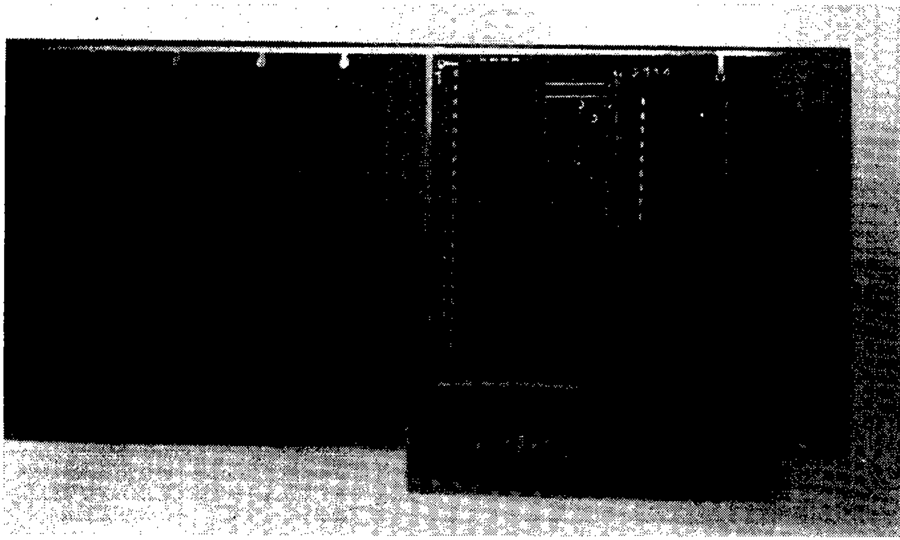


4.8 Anschlußbelegung eines Steckplatzes

4.3 Die I/O Karte 6526

Die in Abbildung 4.9 gezeigte Karte wurde als 6522 Erweiterungskarte für den APPLE II entworfen. Für den C-64 wird der Baustein 6526 verwendet.
Sie wurde dort mit großem Erfolg zur Ansteuerung von Druckern, AD–DA Wandlern und Musikbausteinen verwendet. Auf der linken

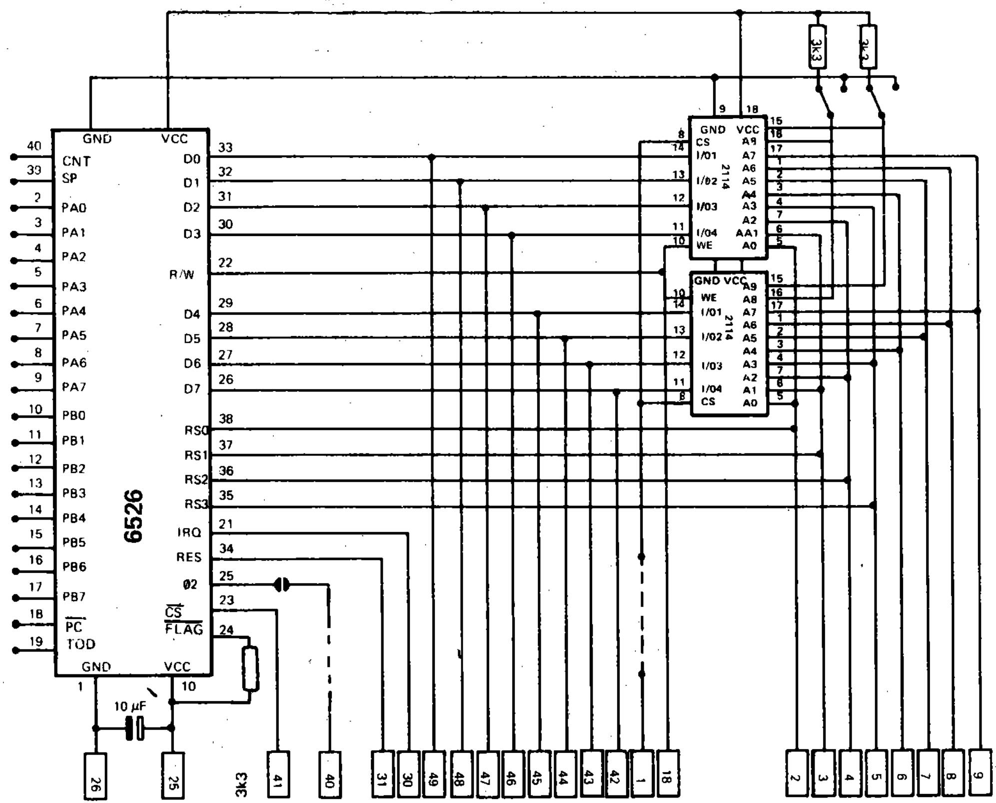
Seite der Platine ist freier Platz zum Aufbau von Versuchsschaltungen. Auf jeder Platine ist 1/4k RAM vorhanden, das über I/O Select dekodiert ist. Die Register des 6526 werden durch DEVICE SELECT ausgewählt. Das Schaltbild zeigt Abbildung 4.10.



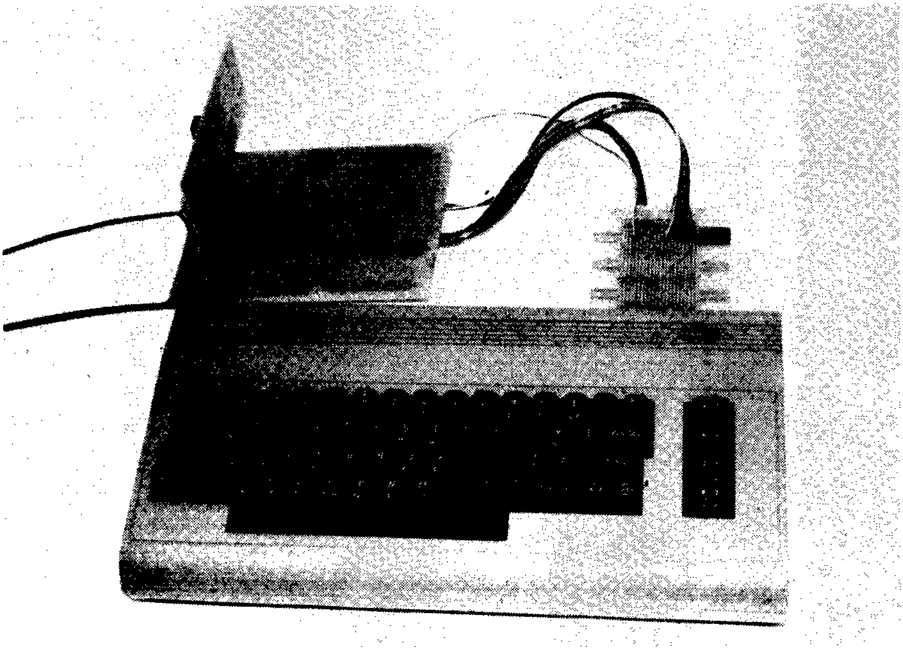
4.9 I/O Karte 6526

Die Zusammenschaltung von C-64, Ein-/Ausgabekarte und 6526 Karte zeigt Abbilding 4.11.

Die Leitungen zwischen Expansion Port und Platine sollten möglichst kurz sein. Auf der Steckplatine am C-64 ist auf der rechten Seite der DIL-Schalter, mit welchem der RAM-Bereich abgeschaltet wird. In den Steckplatz 4 ist eine 6526 Karte gesteckt. Eine Anwendung dieser Karte zeigt der nächste Abschnitt.



4.10 Schaltbild der I/O Karte

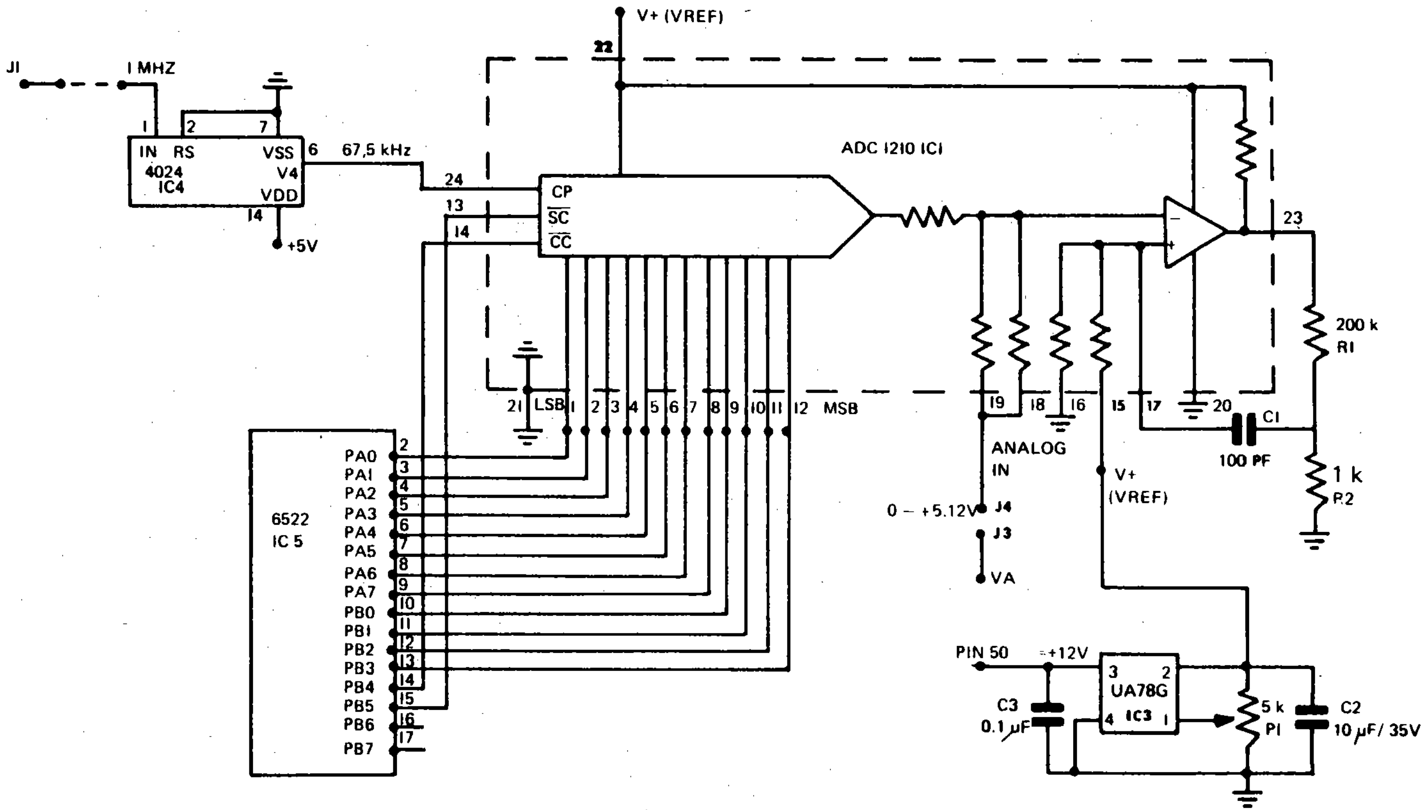


4.11 Zusammenschaltung C-64, Ein-/Ausgabeplatine und 6526 Karte

4.4 Anschluß des 12-Bit Analog-Digital Wandlers ADC 1210

In vielen Fällen ist die Auflösung eines 8-Bit Wandlers zu gering. Ein 12-Bit Wandler bietet dann schon eine bessere Auflösung. Hätte man in dem Beispiel mit der Temperaturmessung statt des 8-Bit Wandlers einen 12-Bit Wandler benutzt, so hätte man die Temperatur auf 5/100°C genau messen können. Das Schaltbild zeigt Abbildung 4.12.

Die Ausgänge des Wandlers sind mit den Toren A und B verbunden. Von den vier noch freien Leitungen des Tores B werden PB4 zum Starten des Wandlers ($\overline{SC}$) und PB5 zur Feststellung des Endes der Wandelzeit ($\overline{CC}$) benutzt. Die Wandlung einer Spannung in einen Digitalwert erfolgt auch bei diesem Wandler nach der Methode der sukzessiven Approximation. Der dafür benötigte Takt wird über einen Frequenzteiler 4024 aus dem Prozessortakt Ø2 gewonnen. Die positive Referenzspannung legt auch gleichzeitig den Eingangsspannungsbereich



4.12 Programm zur Messwerterfassung mit dem ADC 1210

fest. Mit einem einstellbaren Spannungsregler wird VREF=5.12 Volt eingestellt. Damit kann eine Spannung von 0 bis 5.12 Volt in einen Zahlenwert gewandelt werden.

Das Programm zeigt Abbildung 4.13. Die Wandlung wird durch einen Tastendruck gestartet. Die internen Timer sind so programmiert, daß exakt ein Messwert pro Sekunde aufgenommen wird.

Das Assemblerprogramm zur Messwerterfassung ist in dem RAM auf der 6526 Karte gespeichert. Der Wandler ist auf dem Experimentierfeld der Karte aufgebaut.

BASIC:

```
100 DIM MA(500)
110 MS=49156:MZ=49165
120 MW=49196:NM=49232
130 MB=49406
200 PRINT""
210 INPUT"MESSUNG BEGINNT NACH RETURN";A$
220 I=0
230 SYS MS:SYS MZ: SYS MW
240 GOSUB 300
250 SYS NM:SYS MW:GOSUB 300:GOTO 250
300 M=PEEK(MB+1)*256+PEEK(MB)
310 PRINT M
320 MA(I)=M:I=I+1:RETURN
```

ASSEMBLER:

```
PORTA    EQU $80C0
PORTB    EQU $80C1
DDRB     EQU $80C3
T1       EQU $80C4
T2       EQU $80C6
ICR      EQU $80CD
CRA      EQU $80CE
CRB      EQU $80CF

AUXA     EQU $C0FE
```

```
                              ORG $C000
C000: 205DC0                  JSR MAIN
C003: 00                      BRK


C004: A920    INIT            LDA #$20
C006: 8DC380                  STA DDRB
C009: 8DC180                  STA PORTB
C00C: 60                      RTS

C00D: A9C4    START           LDA #$C4
C00F: 8DC480                  STA T1
C012: A909                    LDA #$09
C014: 8DC580                  STA T1+1
C017: A9C8                    LDA #$C8
C019: 8DC680                  STA T2
C01C: A900                    LDA #$00
C01E: 8DC780                  STA T2+1
C021: A947                    LDA #$47
C023: 8DCF80                  STA CRB
C026: A907                    LDA #$07
C028: 8DCE80                  STA CRA
C02B: 60                      RTS

C02C: A900    MW              LDA #$0
C02E: 8DC180                  STA PORTB
C031: A920                    LDA #$20
C033: 8DC180                  STA PORTB
C036: ADC180  M1              LDA PORTB
C039: 2910                    AND #%00010000
C03B: D0F9                    BNE M1
C03D: ADC180                  LDA PORTB
C040: 290F                    AND #$0F
C042: 490F                    EOR #$0F
C044: 8DFFC0                  STA AUXA+1
C047: ADC080                  LDA PORTA
C04A: 49FF                    EOR #$FF
C04C: 8DFEC0                  STA AUXA
C04F: 60                      RTS

C050: A902    NMW             LDA #$02
C052: 8DCD80                  STA ICR
C055: ADCD80  M2              LDA ICR
```

```
C058: 2902                  AND #%00000010
C05A: F0F9                  BEQ M2
C05C: 60                    RTS

C05D: 2004C0 MAIN           JSR INIT
C060: 200DC0                JSR START
C063: 202CC0 MA             JSR MW
C066: 208EC0                JSR OUTA
C069: 2050C0                JSR NMW
C06C: 18                    CLC
C06D: 90F4                  BCC MA

             BSOUT          EQU $FFD2
             AUX            EPZ $F8


C06F: 85F8   PRTBYT         STA AUX
C071: 4A                    LSR
C072: 4A                    LSR
C073: 4A                    LSR
C074: 4A                    LSR
C075: 2080C0                JSR PRT
C078: A5F8                  LDA AUX
C07A: 2080C0                JSR PRT
C07D: A5F8                  LDA AUX
C07F: 60                    RTS

C080: 290F   PRT            AND #$0F
C082: C90A                  CMP #$0A
C084: 18                    CLC
C085: 3002                  BMI P
C087: 6907                  ADC #$07
C089: 6930   P              ADC #$30
C08B: 4CD2FF                JMP BSOUT

C08E: ADFFC0 OUTA           LDA AUXA+1
C091: 206FC0                JSR PRTBYT
C094: ADFEC0                LDA AUXA
C097: 206FC0                JSR PRTBYT
C09A: A90D                  LDA #$0D
C09C: 4CD2FF                JMP BSOUT
PHYSICAL ENDADDRESS: $C09F
```

FORTH:

```
SCR # 27
( ADC 1210                      29.11.EF)
HEX
80C0 CONSTANT PORTA
80C1 CONSTANT PORTB
80C3 CONSTANT DDRB
80C4 CONSTANT T1
80C6 CONSTANT T2
80CD CONSTANT ICR
80CE CONSTANT CRA
80CF CONSTANT CRB
: INIT 20 DUP DDRB C! PORTB C! ;

: STI 9C4 T1 ! C8 T2 ! 47 CRB C!
  07 CRA C! ;

DECIMAL ;S

SCR # 28
( ADC 1210 CNTD                 29.11.EF)
CODE ST HEX 0 # LDA, PORTB STA,
  20 # LDA, PORTB STA, NOP,
  BEGIN, PORTB LDA, 10 # AND, 0=
  UNTIL, DEX, DEX, PORTA LDA,
  BOT STA, PORTB LDA, 0F # AND,
  BOT 1+ STA, NEXT JMP, END-CODE

CODE TI 02 # LDA, ICR STA,
  BEGIN, ICR LDA, 02 # AND, 0=
  NOT UNTIL, NEXT JMP, END-CODE

: MESSEN ST . BEGIN TI ST .
  ?TERMINAL UNTIL ;

DECIMAL ;S
OK
```

4.13 Programm zur Messwerterfassung mit dem ADC 1210

Sowohl BASIC als auch FORTH benutzen Assembler Unterprogramme. Das Unterprogramm INIT setzt Bit 5 des Tores B als Ausgang. START setzt und startet die Timer. Timer 1 ist so programmiert, daß nach jeweils 2.5ms ein Nulldurchgang auftritt. Diese Zeit ist nochmals im Timer 2 so untersetzt, daß dort alle Sekunden ein Nulldurchgang auftritt. Zu diesem Zeitpunkt wird ein Messwert aufgenommen. Dies geschieht im Programmteil MW. Der Wandler wird durch einen Impuls an PB4 gestartet. Danach wird PB5 abgefragt, ob das Ende der Wandelzeit erreicht ist. Dann wird der Messwert in den Zellen C0FE und C0FF gespeichert. Der Ausgangscode des ADC1210 ist ein komplementärer Code. Bei Vollauschlag liegt 00, bei Null Volt am Eingang FFF am Ausgang an. Deshalb wird der Wert durch EOR mit $FF umgewandelt. Im Programmteil NMW wird aus den nächsten Zeitpunkt zur Messwertaufnahme gewartet. Dazu wird Bit 2 im Register ICR gelöscht. Bei jedem Nulldurchgang des Timers 2 wird dieses Bit auf Eins gesetzt. Das Programm wartet drauf in einer Warteschleife.

In BASIC werden die einzelnen Programmteile durch SYS Befehle aufgerufen. Die Messwerte werden auf den Bildschirm ausgegeben und gleichzeitig in einem Zahlenfeld MA gespeichert. Das Programm muß mit STOP abgebrochen werden.

In Assembler wird das Programm ab $C000 gestartet. Die Messwerte werden nur auf den Bildschirm ausgegeben.

In FORTH wird durch das Wort INIT das Tor B initialisiert. STI startet die Timer. Das Wort ST löst die Wandlung eines Messwertes aus und legt ihn auf dem Stapel ab. TI wartet auf den nächsten Nulldurchgang von Timer 2. Das Wort MESSEN zeigt eine Reihe von Messwerten auf dem Bildschirm an.

Bei dieser Art der Programmierung kann ein Messwert fehlen, wenn gerade ein Prozessor Interrupt anliegt. Deshalb sollte für eine exakte Messwertreihe der Interrupt des Prozessors abgeschaltet oder ebenfalls mit Interrupts gemessen werden.

Schlußbemerkung:

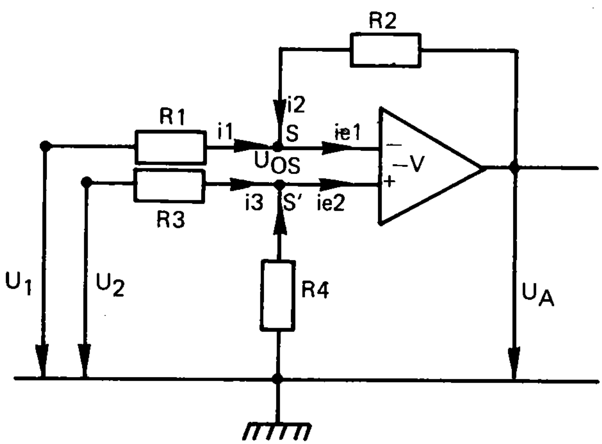
Es konnte nur ein kleiner Teil der Möglichkeiten zur Messwerterfassung mit einem Rechner gezeigt werden. Viele Anwendungen, wie zum

Beispiel die Berührungslose Drehzahl oder Längenmessung sind nicht beschrieben worden. Aber solche Aufgaben lassen sich anhand beschriebener Beispiele lösen. Gerade in der Messwerterfassung können keine fertigen Lösungen gegeben werden. Alle Beispiele sind Lösungsvorschläge, die den jeweiligen Messaufgaben angepasst werden müssen.

A Grundlagen der Operationsverstärker

Grundlagen der Operationsverstärker

Der Begriff des Operationsverstärkers stammt aus der Analogrechentechnik. In dieser Rechentechnik wird nicht mit Zahlenwerten, sondern mit analogen Spannungen gerechnet. Der Operationsverstärker ist ein Gleichspannungsverstärker mit sehr hoher innerer Verstärkung. Dieser kann, je nach äußerer Beschaltung, als Summierer, Integrator, Inverter oder als Differrentiator verwendet werden. Vor etwa 30 Jahren waren diese Operationsverstärker aus Röhren aufgebaut und benötigten zum Betrieb hohe Spannungen. Die Verlustleistung war sehr hoch.



C-1 Grundschaltung eines Differenzverstärkers

Die heutigen integrierten Operationsverstärker haben eine sehr kleine Leistungsaufnahme und arbeiten meist mit Ausgangsspannungen von ± 15 Volt. Im Gegensatz zu den Röhrenverstärkern, die nur einen Eingang hatten, haben die integrierten Verstärker Differenzeingänge. Die Abbildung C-1 zeigt einen mit den Widerständen R1 bis R4 beschalteten Verstärker. Für diesen sollen nun die Grundgleichungen abgeleitet werden. Im folgenden wird statt des Wortes Operationsverstärker die Bezeichnung OPAMP gebraucht.

Folgende Annahmen werden gemacht:

Die Ausgangsspannung UA muß Null sein, wenn U1 und U2 ebenfalls Null sind. In diesem Fall ist

$$UA = -V \cdot UOS$$

Da V, die differentielle Verstärkung, sehr groß ist, muß UOS Null sein. Die Punkte S und S' liegen auf gleichen Potential, sie sind virtuell miteinander verbunden. S wird auch als Summenpunkt des Verstärkers bezeichnet.

In den Knoten S und S' addieren sich die Ströme i1, i2 und ie1 bzw. i3, i4 und ie2. Für einen idealen OPAMP sind die Eingangsströme ie1 und ie2 Null. Damit ergibt sich:

$$i1 + i2 = 0 \quad \text{und}$$
$$i3 + i4 = 0$$

Für die Masche R4, R3 und U2 gilt:

$$i_4 R - i\ R + U = 0$$
$$i_4 \cdot (R_4 + R_3) + U_2 = 0 \qquad (1)$$

und nach i4 aufgelöst:

$$i_4 = -\frac{U_2}{R_4 + R_3} \qquad (2)$$

Für die Masche UA, R2, R1 und U1 gilt:

$$-U_A + i_2 \cdot R_2 - i_1 \cdot R_1 + U_1 = 0$$

$$i_2 (R_1 + R_2) + U_1 - U_A = 0 \qquad (3)$$

und nach i2 aufgelöst:

$$i_2 = \frac{U_A - U_1}{R_1 + R_2} \qquad (4)$$

Durch eine dritte Gleichung mit der Masche UA, R2 und R4 (mit U0S=0) wird ein Zusammenhang zwischen i2 und i4 aufgestellt.

$$-U_A + i_2 R_2 - i_4 R_4 = 0 \qquad (5)$$

In diese Gleichung wird nun i2 und i4 eingesetzt und nach UA aufgelöst.

$$-U_A + \frac{U_A - U_1}{R_1 + R_2} \cdot R_2 + \frac{U_2}{R_3 + R_4} R_4 = 0$$

$$U_A \cdot \left(-1 + \frac{R_2}{R_1 + R_2}\right) - U_1 \cdot \frac{R_2}{R_1 + R_2} + U_2 \cdot \frac{R_2}{R_3 + R_4} = 0$$

$$U_A = -U_1 \cdot \frac{R_2}{R_1} + U_2 \cdot \frac{R_4}{R_3 + R_4} \cdot \frac{R_1 + R_2}{R_1} \qquad (6)$$

$$U_A = -U_1 \cdot \frac{R_2}{R_1} + U_2 \cdot \frac{R_4}{R_1} \cdot \frac{R_1 + R_2}{R_3 + R_4}$$

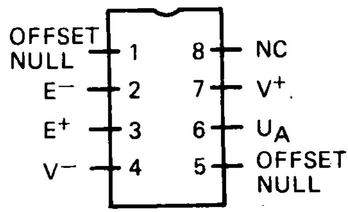
Wird in der letzten Gleichung nun R1 = R3 und R2 = R4 gesetzt,

ergibt sich

$$U_A = \frac{R_2}{R_1}(U_2 - U_1) \qquad (7)$$

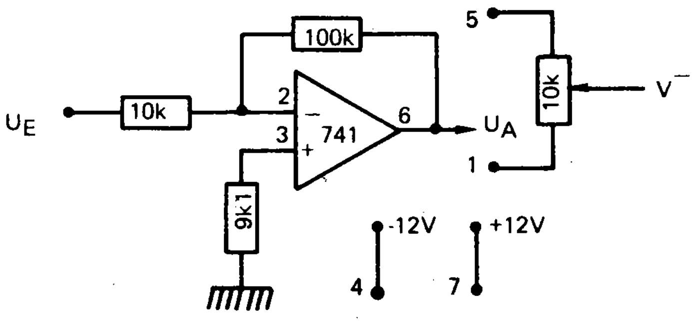
Das Verhältnis R2/R1 ist der Verstärkungsfaktor, mit welchem die Spannungsdifferenz U2—U1 verstärkt wird.

Mit dem OPAMP 741, einem sehr weit verbreiteten und billigen Bauelement werden nun einige Grundschaltungen aufgebaut. Die Anschlußbelegung zeigt Abbildung C-2.



C-2 Anschlußbelegung 741

Wird in Gleichung (7) U2 Null, so hat man einen invertierenden Verstärker. Abbildung C-3 zeigt einen Verstärker mit dem Verstärkungsfaktor 10.



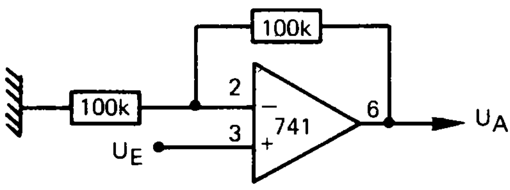
C-3 Invertierender Verstärker

Der Widerstand von 9,1k vor dem positiven Eingang dient zur Minimisierung der Offsetspannung. Das ist die Spannung, die am Ausgang anliegt, obwohl die Eingangsspannung Null Volt ist. Zusätzlich kann beim 741 ein Potentiometer zur Einstellung der Offset Spannung hinzugeschaltet werden.

Die Abbildung C-4 zeigt einen nicht invertierenden Verstärker. Für die Ausgangsspannung gilt:

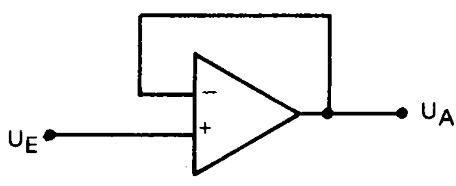
$$U_A = U_E \cdot \left(1 + \frac{R_2}{R_1}\right) \tag{8}$$

Die Verstärkung der Schaltung in Abbildung C-4 ist somit 2.



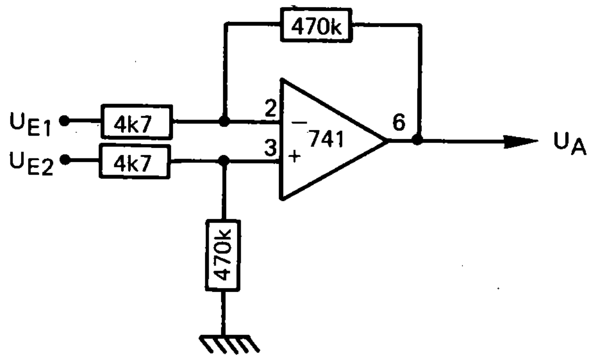
C-4 Nicht inventierender Verstärker

Wird R2 zu Null gemacht, so erhält man die Schaltung eines Spannungsfolgers. Die Verstärkung ist 1 und da in den Eingang keine (nur geringe) Ströme fliessen, ist der Eingangswiderstand sehr hoch.



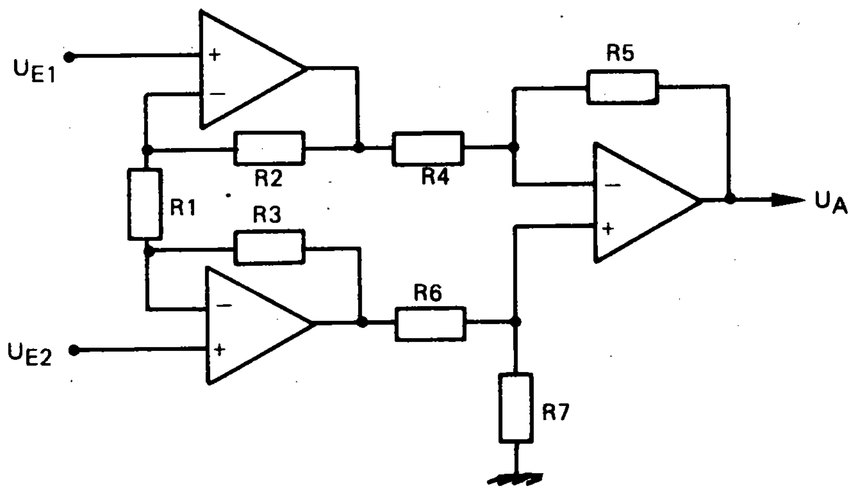
C-5 Spannungsfolger

In Abbidlung C-6 ist nun ein Differenz-Verstärker gezeigt. Mit der angegebenen Dimensionierung ist der Verstärkungsfaktor 100.



C-6 Differenzverstärker

Diese Schaltung hat den Nachteil, daß die Eingangswiderstände der beiden Eingänge klein und unterschiedlich sind. Eine bessere Lösung ist der Instrumentationsverstärker. Die Schaltung zeigt Abbildung C-7.

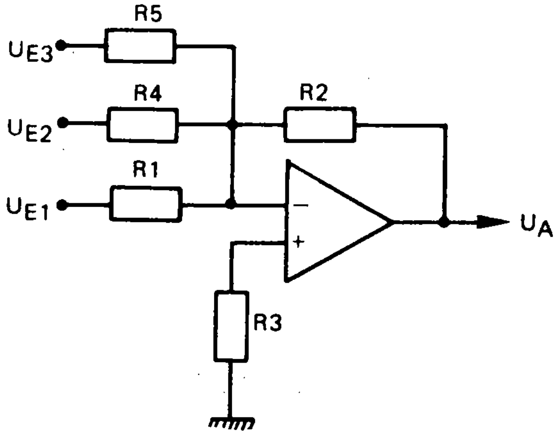


C-7 Instrumentationsverstärker

Die erste Stufe des Verstärkers ist nicht invertierend. Wenn R2 = R3 ist, dann ist die Verstärkung

$$U_A = (U_{E2} - U_{E1})\left(1 + \frac{2R_2}{R_1}\right) \quad (9)$$

Die zweite Stufe ist ein Differenzverstärker. Seine Verstärkung hängt von der Dimensionierung der Widerstände ab. Wenn R4=R5=R6=R7 ist, so ist (9) die Gesamtverstärkung des Instrumentationsverstärkers. Zum Aufbau verwendet man am besten einen 4-fach OPAMP, wie zum Beispiel LM324, RC4138 oder TL064C.



C-8 Summierverstärker

Abbildung C-8 zeigt einen Summierverstärker. Die Ausgangsspannung Ua ist:

$$U_A = -\left(U_{E1}\frac{R_2}{R_1} + U_{E2}\frac{R_2}{R_4} + U_{E3}\frac{R_2}{R_5}\right) \qquad (10)$$

Ein Summierverstärker wird dann verwendet, wenn einer Meßspannung eine konstante Spannung überlagert werden muß. Dies ist zum Beispiel dann der Fall, wenn ein Analog-Digital Wandler einen Eingangsspannungsbereich von 0 bis 10 Volt hat, die Meßspannung aber zwischen –5 und +5 Volt schwankt. Zu dieser Spannung wird dann 5 Volt addiert und sie somit an die Eingangsspannung des Wandlers angepasst.

Der in Kapitel 2 verwendete LM3900 ist kein Operationsverstärker im herkömmlichen Sinn. Bei diesem wird nicht die Differenz von

Spannungen, sondern von Strömen in der Eingangsstufe gebildet. Er hat somit ein anderes Verhalten. Sein Vorteil ist, daß er nur mit einer Versorgungsspannung betrieben werden kann, während herkömmliche OPAMPS immer zwei Spannungen ($\pm$15 Volt) benötigen.

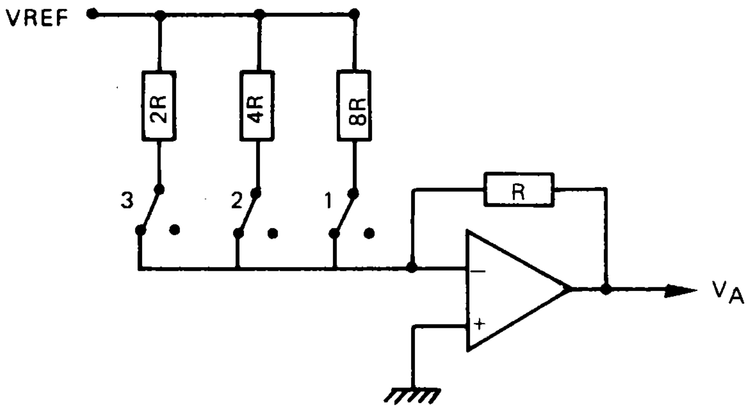
B Grundlagen der A/D- und D/A-Wandlung

Grundlagen der Analog-Digital und Digital-Analog Wandlung

(Nach Seminar-Unterlagen der Firma Analog-Devices, Inc.)

1. Digital-Analog Wandlung

Die Abbildung D-1 zeigt das Prinzipschaltbild eines 3-Bit Digital-Analog Wandlers.

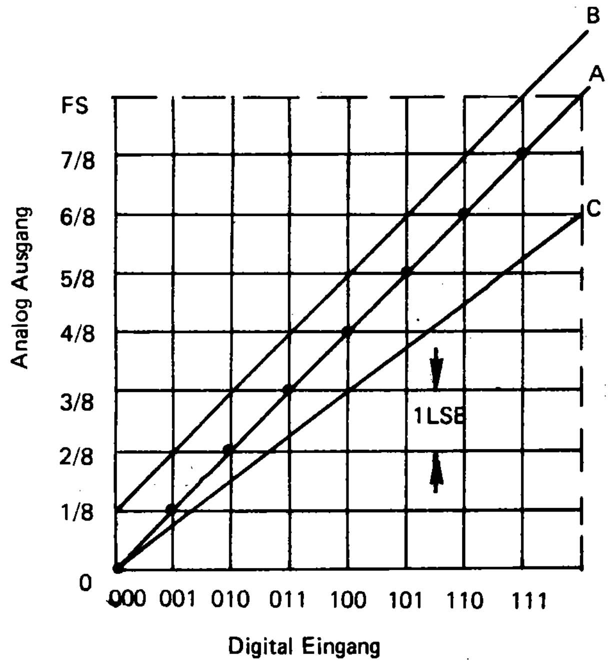


D-1 Prinzipschaltbild eines 3-Bit Digital-Analog Wandlers

Der Operationsverstärker ist mit einem Widerstand R rückgekoppelt. Am Summenpunkt sind über Schalter die Widerstände 2R, 4R und

8R angeschlossen. Ist zum Beispiel der Schalter 3 geschlossen und alle anderen offen, so ist die Verstärkung des Gesamtsystems gleich 1/2. Am Ausgang liegt dann die Spannung VREF/2. Dies entspricht einem Eingangscode von %100.

Sind alle drei Schalter geschlossen, so ist der Eingangswiderstand 8/7*R und die Ausgangsspannung somit 7/8*VREF.



D-2 Ideale Wandler Kennlinie und Kennlinien mit Fehler

Die Abbildung D-2 zeigt die Ausgangsspannung für die digitalen Eingangscode. Die Linie A ist die ideale Kennlinie eines Digital-Analog Wandlers. Die Ausgangsspannung tritt jeweils in Sprüngen von 1LSB (Least Significant Bit) auf. Dies ist die Auflösung eines Wandlers. Bei einem Vollausschlag (FS Full Scale) des Wandlers von 10 Volt ergeben sich dabei folgende Zahlenwerte:

8-Bit Wandler	1LSB=39.1mV
10-Bit Wandler	1LSB=9.77mV
12-Bit Wandler	1LSB=2.44mV

Folgende Fehler können bei Wandlern auftreten:

Nullpunkt (Offset) Fehler

Die Linie B zeigt einen Nullpunkt-Fehler. Die Ausgangsspannung für den Code %000 ist 1/8 VREF. Jeder Punkt der Linie B ist um diesen Betrag verschoben.

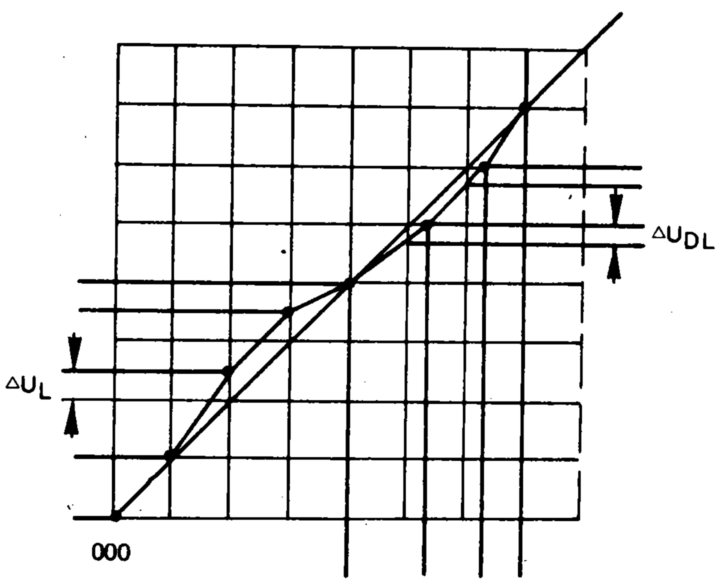
Verstärkungsfehler

Diesen Fehler zeigt Linie C. Die Steigung, und damit die Verstärkung ist ungleich der Steigung der Linie A. Alle Ausgangsspannungen sind um den gleichen Prozentsatz verschieden.

Beide Fehler lassen sich durch eine äußere Beschaltung korrigieren. Dies ist bei den beiden anderen Fehlern nicht möglich.

Liniearitätsfehler

In Abbildung D-3 ist im linken unteren Teil das Auftreten eines Lineartätsfehlers gezeigt.



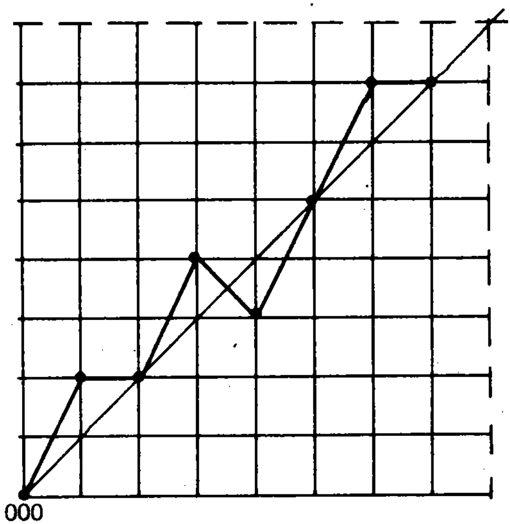
D-3 Linearitätsfehler

Die Ausgangsspannung für den Digitalwert %010 ist um 1/2LSB zu hoch. Dies bewirkt ein Abweichen der Ausgangsspannung von einer geraden Linie. Der Spannungssprung %001 nach %010 ist 1LSB+DUL.

Differentieller Linearitätsfehler

Bei der Differentiellen Linearität wird die Spannungsdifferenz zwischen zwei benachbarten Punkten gemessen. Sind beide gleich 1LSB, so ist der differentielle Linearitätsfehler gleich Null.

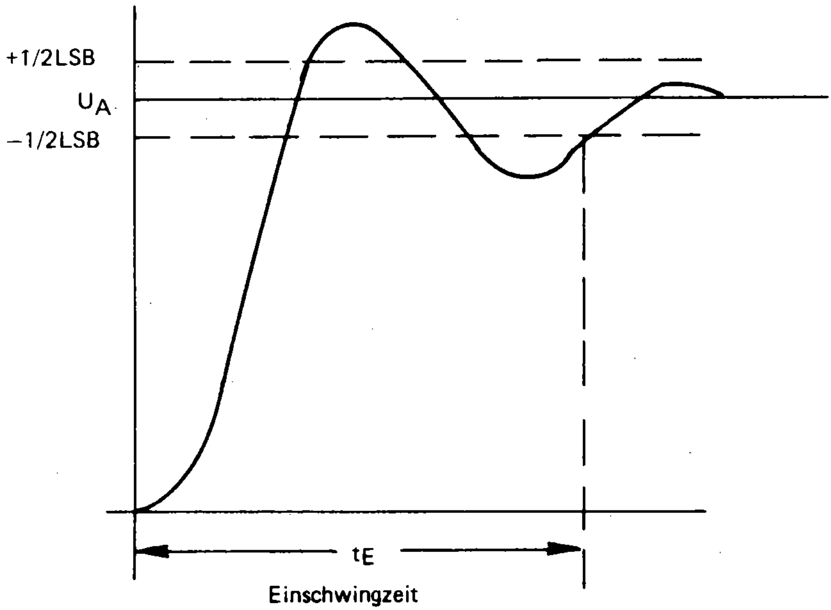
Ein besonders krasses Beispiel zeigt Abbildung D-4.



D-4 Fehler bedingt durch Differentielle Nichtlinearität

Der Ausgangswert für den Code %100 ist kleiner als für den Wert %011. Wird so ein Wandler als Digital-Analog Wandler verwendet, so führt dies zu "missing codes". Bei der Wandlung einer Spannung in Zahlenwerte kann ein bestimmter Zahlenwert nie auftreten.

Die Wandelzeit wird im wesentlichen durch das Einschwingverhalten des Wandlers bestimmt. Eine Möglichkeit die Einschwingzeit anzugeben zeigt Abbildung D-5.



D-5 Einschwingverhalten

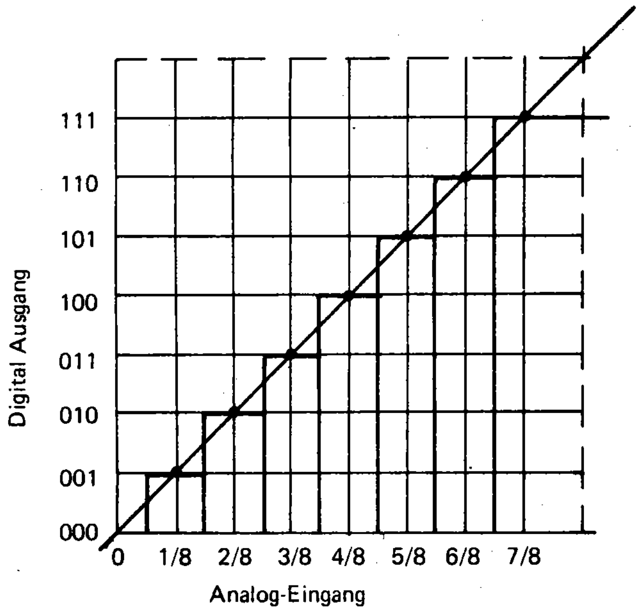
Die Einschwingzeit ist die Zeitdauer vom Beginn der Umsetzung bis zu dem Zeitpunkt, in welcher sich die Ausgangsspannung innerhalb gewisser Fehlerschranken befindet. Wird als Fehlerschranke ± 1/2 LSB angenommen, so muß ein 8-Bit Wandler innerhalb ± 20mV, ein 12-Bit Wandler innerhalb ± 1.25mV einschwingen. Deshalb ist unter Umständen ein 12-Bit Wandler langsamer als ein 8-Bit Wandler, weil seine Spezifikation enger gefasst ist.

Das Einschwingverhalten ist unterschiedlich, ob der Wandler von Null Volt bis zum Vollauschlag oder umgekehrt geschaltet wird.

2. Analog-Digital Wandlung

Die Abbildung D-6 zeigt die ideale Kennlinie eines Analog-Digitalwandlers. Ein digitaler Code am Ausgang wird nicht durch eine einzige Spannung, sondern durch einen kleinen Spannungsbereich erzeugt. Dieser Spannungsbereich wird auch als Code-Breite bezeichnet.

einen idealen Wandler sind die Codebreiten, mit Ausnahme an den Grenzen, überall gleich.



D-6 Ideale Kennlinie eines Analog-Digital Wandlers

Folgende Fehlermöglichkeiten bestehen:

Nullpunktfehler

Bei einem idealen Wandler muß das LSB von 0 auf 1 springen, wenn eine Spannung, die einer halben Codebreite entspricht, an den Eingang gelegt wird. Abweichungen davon sind Nullpunktfehler.

Verstärkungsfehler

Hier gilt die gleiche Definition wie für Digital-Analog Wandler.

Linearitätsfehler

Als Liniaritätsfehler wird die Abbweichung der Code-Mitten von der

geraden Linie bezeichnet. In Abbildung D-6 sind bei einem Linearitätsfehler die Punkte nach rechts oder links verschoben.

Differentieller Linearitätsfehler

Ein solcher Fehler liegt dann vor, wenn die Codebreiten vom idealen Wert 1LSB abweichen. Schrumpft eine Codebreite auf Null zusammen, so stellt dies einen fehlenden Code dar. Dieser Zahlenwert kann nie am Ausgang erscheinen.

Dieser Fehler der "missing codes" ist der einzige Fehler, der nicht korrigiert werden kann. Fehler beim Nullpunkt oder bei der Verstärkung können durch äußere Beschaltung verbessert werden. Liegt ein Linearitätsfehler vor, so kann dieser nach einer Eichung mit einem anderen Wandler durch Software beseitigt werden.

Ein weiterer Fehler kann bei einer Wandlung dann auftreten, wenn sich die Eingangsspannung während der Wandelzeit ändert. Eine Sinusförmige Schwingung der Form

$$u = U \sin(2\pi f t) \quad (11)$$

hat die Steigung

$$u' = 2\pi f U \cos(2\pi f t) \quad (12)$$

Das Maximum dieser Steigung liegt bei

$$u'_{max} = 2\pi f U \quad (13)$$

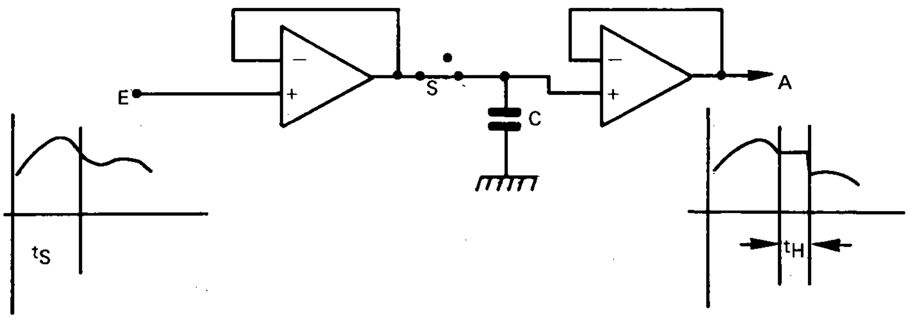
Die Eingangsspannung soll sich nicht mehr als 1/2LSB während der Wandelzeit ändern. Daraus berechnet sich die maximale Frequenz für einen 8-Bit Wandler mit 15 μs Wandelzeit und einer sinusförmigen Eingangsspannung von 10Vss zu

$$f = \approx 42 \text{ Hz} \quad (14)$$

Für einen 12-Bit Wandler mit 35 µs Wandelzeit und der gleichen Eingangsspannung erhält man dagegen

$$f = \approx 1{,}1\ \text{Hz} \qquad (15)$$

Sollen Spannungen mit einer höheren Frequenz gewandelt werden, so müssen Abtast und Halte (Sample und Hold) Verstärker verwendet werden. Ein Beispiel zeigt Abbildung D-7.

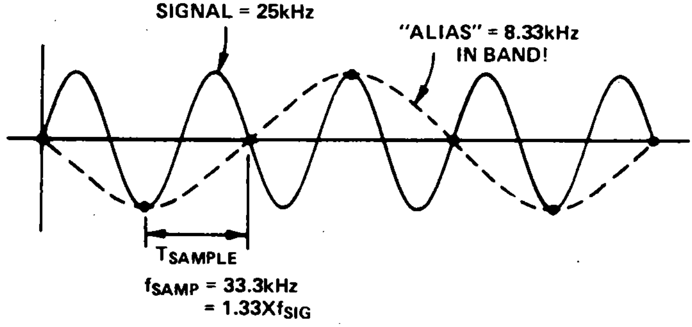


D-7 Abtast und Halteschaltung

Während der Zeit ts liegt die Eingangsspannung am Kondensator C. Wird der Schalter S geöffnet, so bleibt die augenblickliche Spannung am Kondensator erhalten und kann über den Spannungsfolger abgegriffen werden.

Für einen praktischen Einsatz können die integrierten S/H Verstärker AD582 oder AD585 verwendet werden.

Bei der Wandlung von sinusförmigen Spannungen muß auch noch auf folgendes geachtet werden. Nach dem Abtast-Theorem von SHANNON muß eine Schwingung mit der Frequenz f mindestens mit 2*f abgetastet werden, um den Signalverlauf rekonstruieren zu können. Wird mit einer geringeren Frequenz abgetastet, so können sogenannte "ALIAS" Frequenzen auftreten, wie dies in Abbildung D-8 gezeigt ist.



D-8 "ALIAS"-Frequenzen

Der Singalverlauf mit einer Frequenz von 25kHz wird mit 33.3kHz abgetastet. Die Folge der Abtastpunkte stellt eine Frequenz von 8.33kHz dar, die im eigentlichen Signal nicht vorhanden ist.

Notizen

C RS232 Schnittstelle

RS232 Schnittstelle

Dies ist eine weitere Anwendung der 6526 Karte.

Wie der C-64, so verfügen fast alle neueren Heimcomputer und Peripheriegeräte über eine RS232 Schnittstelle. Mit dieser können Rechner untereinander oder mit Zusatzgeräten verbunden werden. Die Datenübertragung erfolgt seriell. Für die Spannungspegel bei der Datenübertragung wurden folgende Werte festgelegt: Spannungen kleiner –3 Volt entsprechen einer logischen Eins, Spannungen größer +3 Volt entsprechen einer logischen Null. Diese Pegel werden heute in den seltensten Fällen eingehalten. Meistens werden die TTL-Pegel verwendet, wobei eine Spannung kleiner 0.8 Volt der logischen Null und Spannungen zwischen 2 und 5 Volt der logischen Eins entsprechen. Aber auch dies trifft nicht immer zu. Bei einem Heimcomputer, der mit CMOS Bausteinen aufgebaut ist, wurden für die Null etwa 0 Volt, für die Eins 10 Volt gemessen.

Schon die Vielfalt der verwendeten Pegel, sowie die freie Verwendung von Steuerleitungen macht das Anschließen von sogenannten RS232 kompatiblen Geräten zu einer echten EDV (Experimentelle Daten Verarbeitung).

Im folgenden wird eine Schaltung beschrieben, das ein Senden und Empfangen von RS232 Signalen ermöglicht. Diese Schaltung sendet Signale als TTL Pegel, kann aber Eingangspegel zwischen –15 und +15 Volt empfangen. In der Abbildung 1 sind die wichtigsten Leitungen und die genormte Steckerbelegung gezeigt.

STECKER NR:		BEZEICHNUNG	
1		SCHUTZERDE	
2	TxD	DATEN AUS	(AUSGANG)
3	RxD	DATEN EIN	(EINGANG)
4	RTS	AUFFORDERUNG ZUM SENDEN	(AUSGANG)
5	CTS	BEREIT ZUM SENDEN	(EINGANG)
6	DSR	DATEN BEREIT	(EINGANG)
7		SIGNAL MASSE	
8		EMPFANGSSIGNAL LEITUNG BEREIT	(EINGANG)
20	DTR	DATEN TERMINAL BEREIT	(AUSGANG)

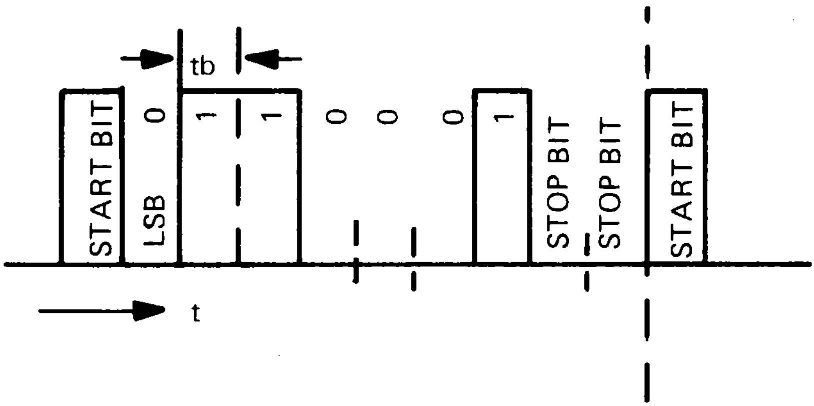
1 Die wichtigsten Signale und die Steckerbelegung einer RS232 Schnittstelle

Über die Leitung 2 (TXD) werden die Daten bitseriell ausgegeben und über die Leitung 3 (RXD) empfangen. Die Leitungen 4 und 5 werden für das Empfangen von Signalen verwendet. Mit Leitung 5 kann die Leitung 4 des Sendegerätes abgefragt werden, ob es bereit ist, Daten zu senden. Beim Senden von Daten wird über die Leitung 6 dem Empfangsgerät angezeigt, daß eine Datenübertragung stattfindet. Vorher kann über die Leitung 20 festgestellt werden, ob das Gerät betriebsbereit ist.

Für die Geschwindigkeit der Datenübertragung sind folgende BAUD-Raten festgelegt: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 und 19200.

Für die serielle Ausgabe auf Drucker werden 300 BAUD verwendet. Diese Geschwindigkeit wird auch bei Modems angewendet. Teilweise werden auch Daten mit 1200 BAUD übertragen. Für die Datenübertragung zwischen einem Terminal und einem Computer werden 9600 oder 19200 BAUD verwendet. Der Fernschreibverkehr bei der deutschen Bundespost erfolgt mit 50 BAUD.

Abbildung 2 zeigt die Reihenfolge der Daten für ein Zeichen.



2 Reihenfolge der Daten für ein Zeichen

Das Signal beginnt mit einem Startbit. In Abbildung 2 werden TTL Pegel angenommen. Dann folgen 7 Datenbit, wobei das LSB auf das Startbit folgt. Das Zeichen wird mit zwei Stopbit abgeschlossen. Dies ist aber nur ein Beispiel.

Bei hohen Datenübertragungsraten wird oft nur ein Stopbit gesendet. Allerdings kann hier vor den Stopbits noch ein Parity-Bit eingefügt sein. Dieses Bit ist Eins, wenn wahlweise die Zahlen der Einsen eine gerade oder eine ungerade Zahl ist (even or odd parity).

Bei einigen Rechnern mit RS232 Schnittstelle kann das Format programmiert werden. Die Angabe 37H2E bedeutet beim TRS-80 Modell 100, daß mit 300 BAUD gesendet und empfangen wird, wobei das Zeichen 7 Bit lang ist, kein Parity-Bit übertragen wird, dafür aber zwei Stopbits.

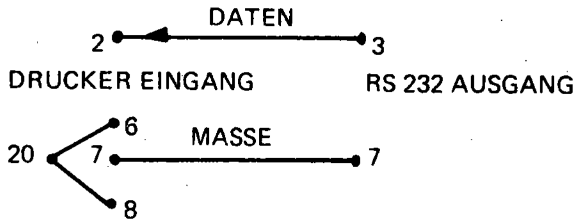
Abbildung 3 zeigt den Aufbau einer RS232 Schnittstelle mit dem Ein-/Ausgabebaustein 6526. Die Ausgabe eines Signals erfolgt über Tor A, Bit 0 (PA0). Über PA1 wird das Signal "DATEN BEREIT" (DSR) an das empfangende Gerät ausgegeben. Werden die Signale bei 3 und 6 abgenommen, so entsprechen die Pegel TTL Ausgängen. Für die

Spannungspegel, wie sie für eine RS232 Schnittstelle notwendig sind, müssen TTL-RS232 Wandler MC1488 nachgeschaltet werden. Die Eingabe erfolgt über einen RS232-TTL Wandler MC1489. Damit ist sichergestellt, daß der folgende TTL Baustein nicht durch hohe Spannungen zerstört wird.

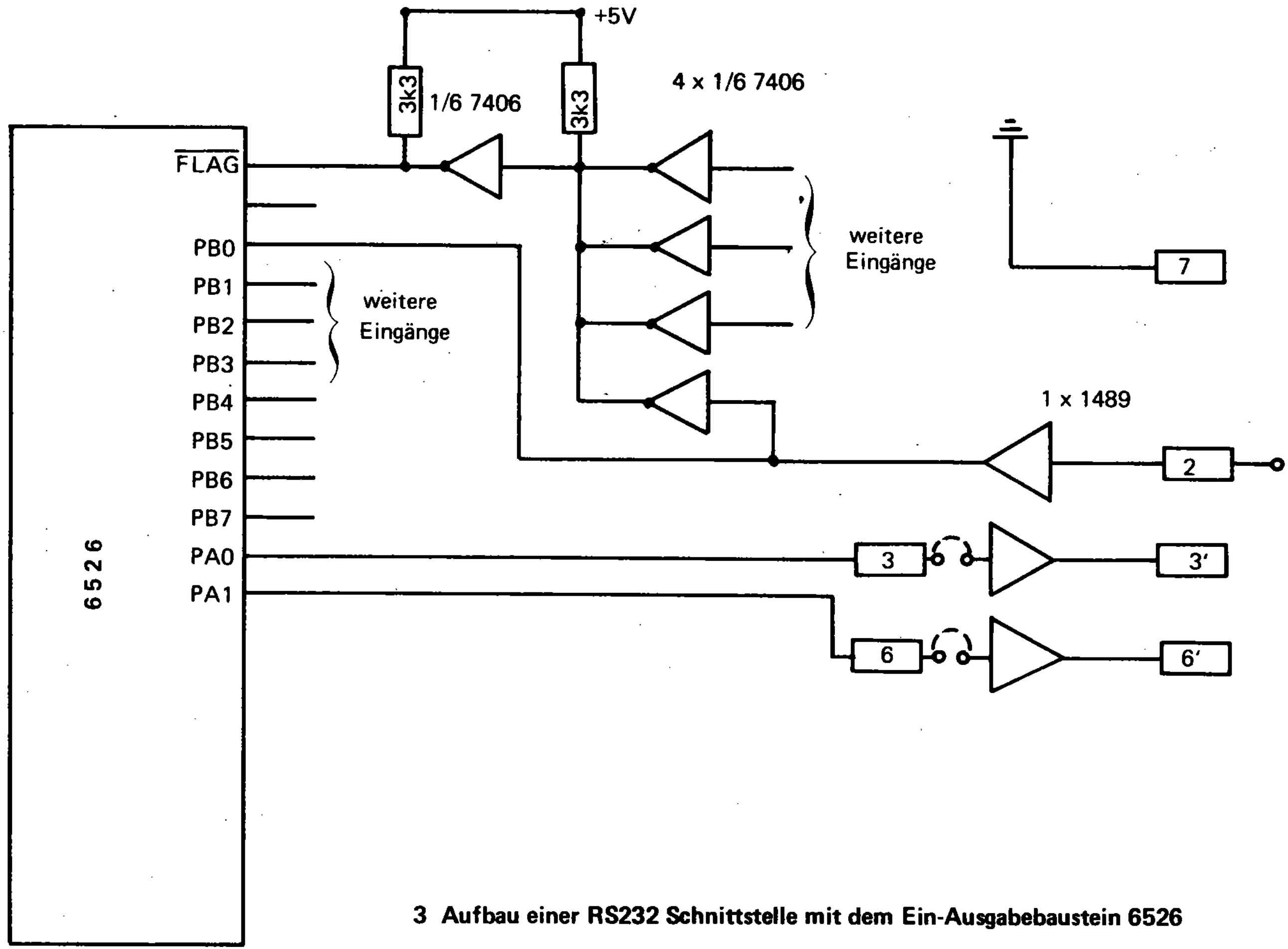
Für den Empfang von Signalen wird nicht das RS232 Protokoll verwendet, sondern eine Interrupt Technik. Damit kann ein ankommendes Signal jederzeit in den Rechner eingelesen werden, obwohl dieser mit anderen Programmen beschäftigt ist.

Wird nur ein Gerät an die Schnittstelle angeschlossen, so kann der 7406 entfallen. PB0 wird direkt mit $\overline{\text{FLAG}}$ verbunden. Bei mehreren Geräten (bis zu vier) werden die Interrupts in einer wired-or Schaltung zusammengefasst und an $\overline{\text{FLAG}}$ gelegt. Die Eingänge werden auf verschiedene PB Eingänge gelegt.

Die Abbildung 4 zeigt den Anschluß eines Druckers an eine RS232 Schnittstelle. Das Signal DATEN TERMINAL BEREIT (DTR) ist am Drucker ein Ausgangssignal. Dieses wird an die Eingänge 6 (DATEN BEREIT) und 8 (EMPFANGSLEITUNG BEREIT) gelegt. Damit ist der Drucker jederzeit in der Lage, über Leitung 2 Daten aufzunehmen und zu drucken.



4 Anschluß eines seriellen Druckers



3 Aufbau einer RS232 Schnittstelle mit dem Ein-Ausgabebaustein 6526

Notizen

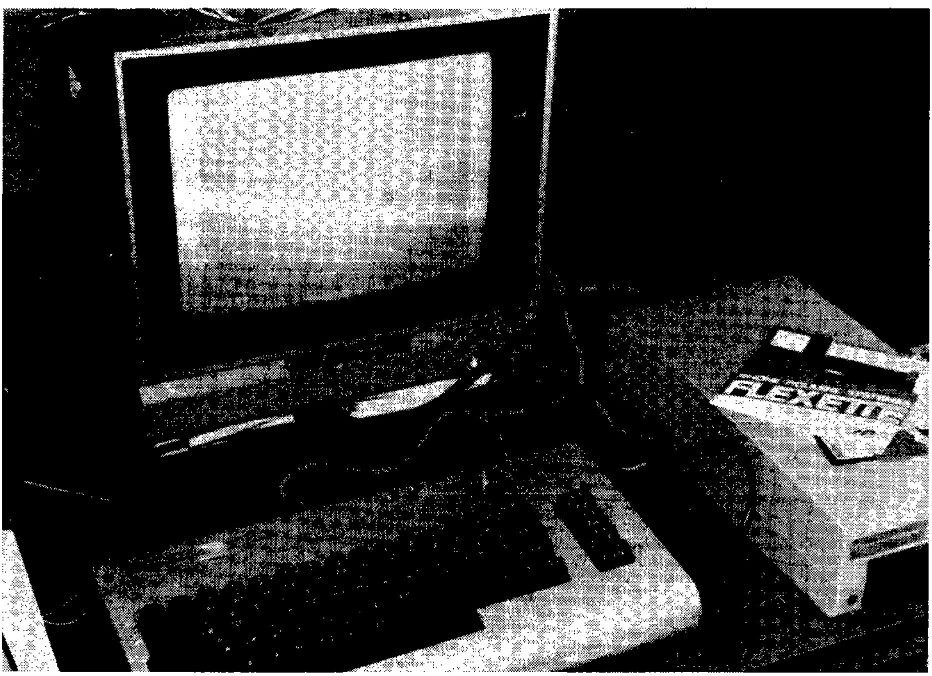
D

RS232 Drucker-anschluß an C-64

RS-232 Druckeranschluß an Commodore 64

Wie schließt man einen seriellen Drucker mit RS-232 Schnittstelle an den Commodore 64 an ?

Jetzt werden Sie sicher denken — das ist ja ganz einfach, der C-64 hat eine serielle RS-232 Schnittstelle, bei der Baudrate Wortlänge, Start-Stop Bits etc. eingestellt werden können und da braucht man jetzt nur noch ein Kabel anzuschließen.



Wenn Sie dies glauben, dann haben Sie sicher noch niemals einen Drucker an einen Personalcomputer angeschlossen. Aus meiner Erfahrung kann ich sagen, daß in 99,5% aller Fälle auf Antrieb überhaupt nichts funktioniert. Ähnlich ist es uns beim Anschluß eines C-64 an einen Qume Sprint 9 gegangen. Der Qume Sprint 9 hat eine RS-232 Schnittstelle und kann über einen sogenannten 2 Leitung Anschluß betrieben werden. Zweileitungsanschluß heißt hier: Für die Datenübertragung sind zwei Leitungen erforderlich. TRANSMITTED DATA auch SIGNAL GROUND. Der C-64 sendet kein normgerechtes RS-232 Signal, sondern ein RS-232 ähnliches Signal mit TTL-Regeln. Dieses Signal ist zum Betrieb der neusten Drucker mit RS-232 Schnittstelle und Modems völlig ausreichend.

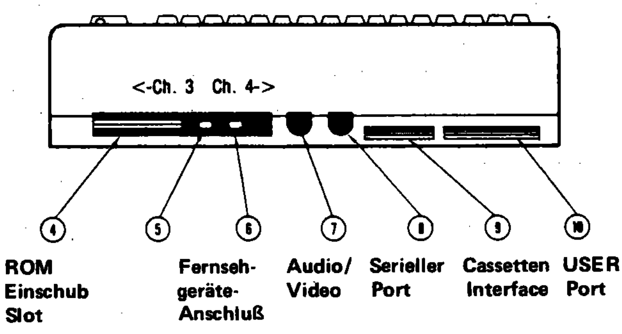
Nachfolgend wollen wir Ihnen einige Tips und Hinweise geben, wie Sie einen RS-232 Drucker oder Modem an Ihren C-64 anschließen können.

Ich habe festgestellt, daß die meisten Typenraddrucker heute mit serieller RS-232 Schnittstelle angeboten werden:

1. Qume Drucker: SPRINT 9
2. BROTHER DRUCKER

Weiterhin ist der RS-232 Betrieb insofern von großer Bedeutung, da die Personalcomputer Communication über Telefon und Satelit in den nächsten Jahren sich noch weiter verbreiten wird.

RS-232 Schnittstellen am C-64



An der Rückseite des C-64 finden wir insgesamt 2 RS-232 Schnittstellen

1. Den RS-232 seriellen Bus
2. Die RS-232 Schnittstelle im User Port.

Die serielle RS-232 Schnittstelle dient zum Anschluß von Disketten der Baureihe VIC 1541 und Graphikdruckern VIC 1525.

Bis zu 5 verschiedene Geräte (z. B. 4 Disks und 1 Drucker) können an diesen RS-232 Bus angeschlossen werden. Der Bus verhält sich ähnlich wie ein IEE Bus, jedoch werden die Daten nicht parallel, sondern seriell ausgegeben und empfangen. Jedem Gerät wird eine logische Gerätenummer zugewiesen, unter der es angesprochen wird und vom C-64 als Empfänger (z. B. Drucker) oder Sender (z. B. Modem oder Disk) geschaltet werden kann. Der Drucker kann unter Device #4 und 5 und die Disk unter Device # 8 angesprochen werden. Dieser serielle Bus eignet sich für einen Druckeranschluß nicht besonders.

Die RS-232 Schnittstelle im C-64 User Port

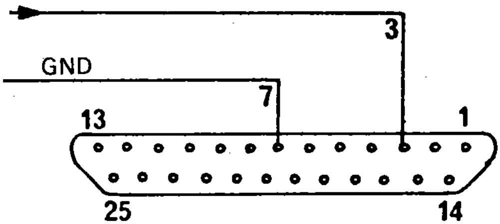
Der C-64 verwendet für den Datenverkehr mit der Außenwelt zwei recht neuartige und leistungsfähige Interface Bausteine.

Einer der beiden CIA 6526 Complex Interface Adapter wurde zum Aufbau der bidirektionalen RS-232 Schnittstelle verwendet. Von den beiden 8 Bit Ports um 6526 werden alle Anschlüsse des Port B und ein Anschluß von Port A sowie ein Flag Anschluß für die RS-232 Schnittstelle belegt. Eine genaue Belegung der Anschlüsse können Sie dem Schaltbild im Commodore Programmers Reference Guide entnehmen.

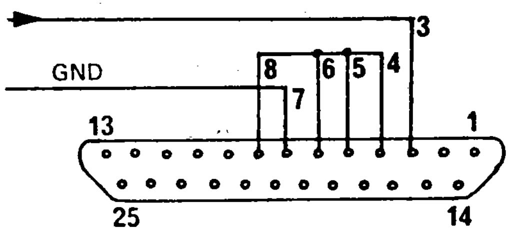
Die Belegung des USER Ports sieht wiefolgt aus:

PB0	Empfangsdaten	Pin C	Eingang
PB1	Sendeteil einschalten	Pin D	Ausgang
PB2	Terminal betriebsbereit	Pin E	Ausgang
PB3	Ankommender Ruf	Pin F	(f. Modembetr.)
PB4	Empfangssignalpegel	Pin H	(Eingang)
PB5	Frei	Pin J	
PB6	Sendebereitschaft	Pin K	(Eingang)
PB7	Betriebsbereitschaft	Pin L	(Eingang)
Flag 2	Empfangsdaten	Pin B	(Eingang)
PA2	Sendedaten	Pin M	(Ausgang)
GND	Gehäuse Erde	Pin A	
GND	Signal Masse	Pin N	

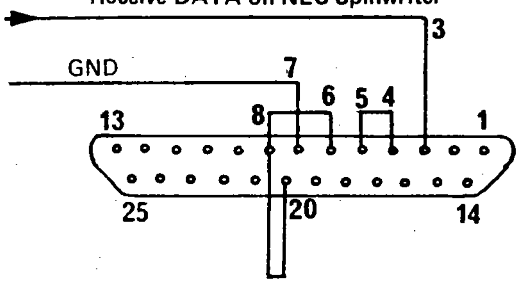
Receive data on DEC-writer



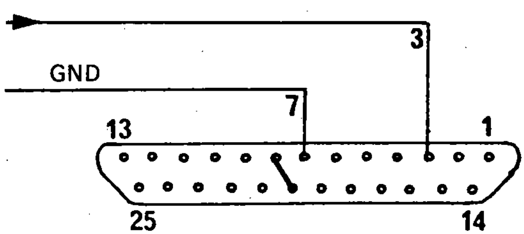
Receive data on Brother HR-15



Receive DATA on NEC Spinwriter



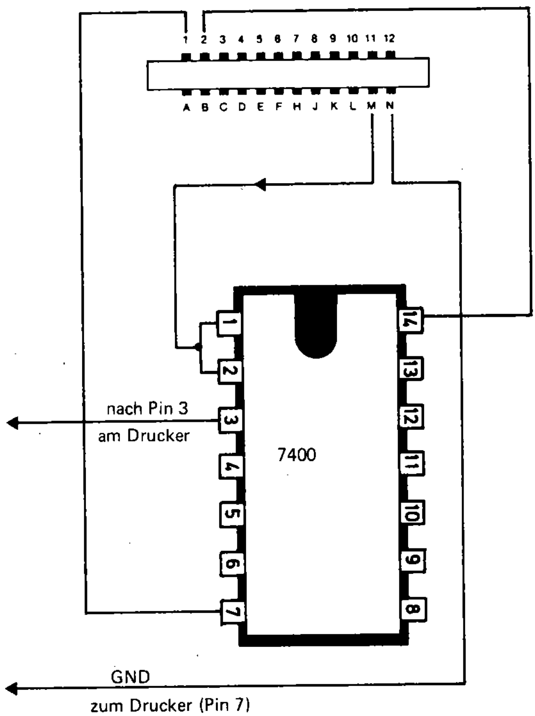
Receive data on Qume Sprint 9



Anschluß M und N am Commodore 64 User Port werden jetzt mit den Anschlüssen 3 und 7 an den Druckern, wie oben beschrieben verbunden.

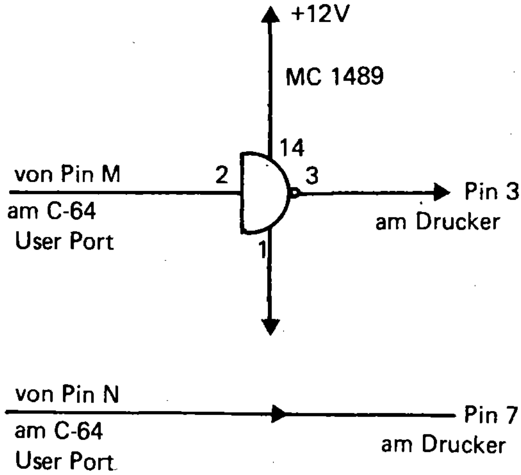
Da der Commodore 64 jedoch ein invertiertes Signal am Ausgang Pin M (PA 2) ausgibt, muß dieses vorher einen Inverter dazwischen geschaltet werden. Der Einfachheit halber kann man hier einen SN 7400 oder SN 7404 auf die Steckerleiste löten und an die Stromversorgung am User Port anschließen.

User Port von hinten gesehen.



Um die Drucker mit nur zwei Leitungen betreiben zu können, müssen die entsprechenden Brücken am Stecker (am Drucker) angebracht werden. Wer Handshake Betrieb (Quittungsbetrieb) machen möchte, kann auch alle zugehörigen Leitungen verbinden.

Eine Invertierung ist jetzt nicht mehr nötig, da der MC 1489 selbst eine Invertierung durchführt.



E Kleine Steckerkunde für den C-64

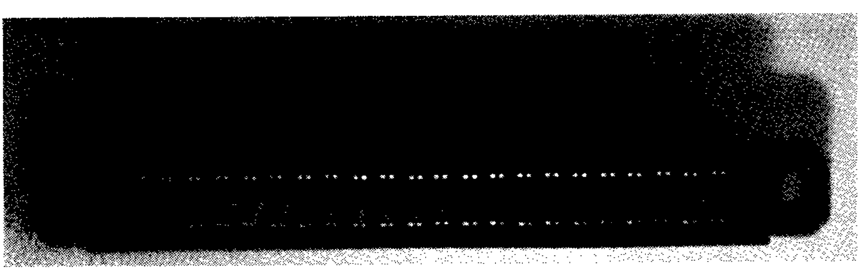
Kleine Steckerkunde für den Commodore 64

Für Erweiterungsexperimente und Druckeranschlüsse werden immer wieder Stecker gebraucht. Der Neuling unter den Anwendern kann sich meist nicht vorstellen, wie die notwendigen Stecker bzw. Sockel aussehen, noch weniger weiß er, wo er sie beziehen kann.

Entsprechend unserer Information "Rund um den Commodore-64" erfolgt hier eine kurze Übersicht mit Bild, Anwendung und Beschreibung.

Stecker für den Modulsteckplatz

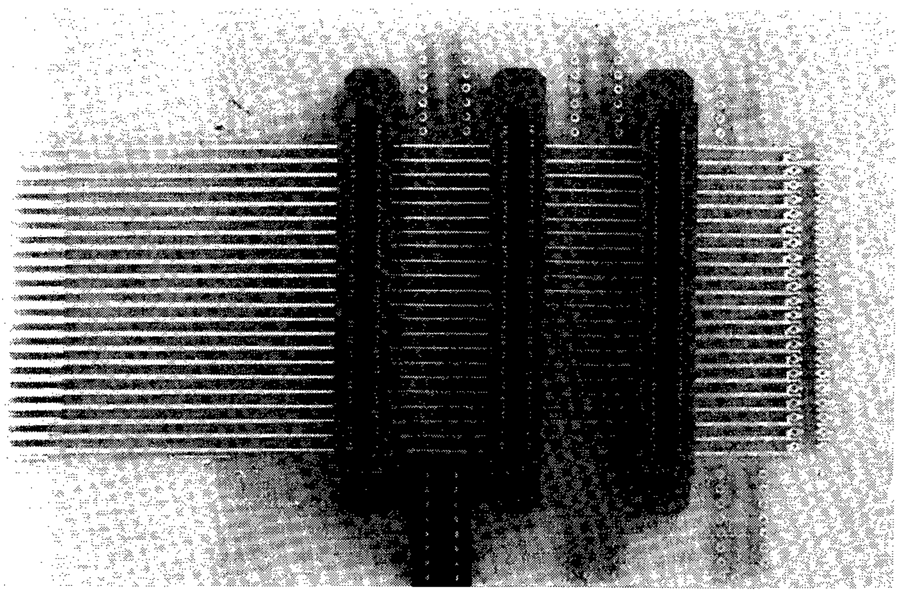
Der Modulsteckplatz wird von einem 44-poligen Sockel gebildet. Der Sockel sieht wie folgt aus:



Sockel (44-polig) von oben gesehen

Sie erkennen diesen Sockel, wenn Sie hinten rechts (von vorne gesehen)

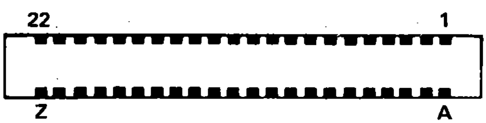
in den Modul-Steckplatz hineinsehen. In diesen Sockel können Sie jetzt eine Expansionsplatine oder eine Experimentierplatine oder verschiedene Erweiterungskarten einstecken.

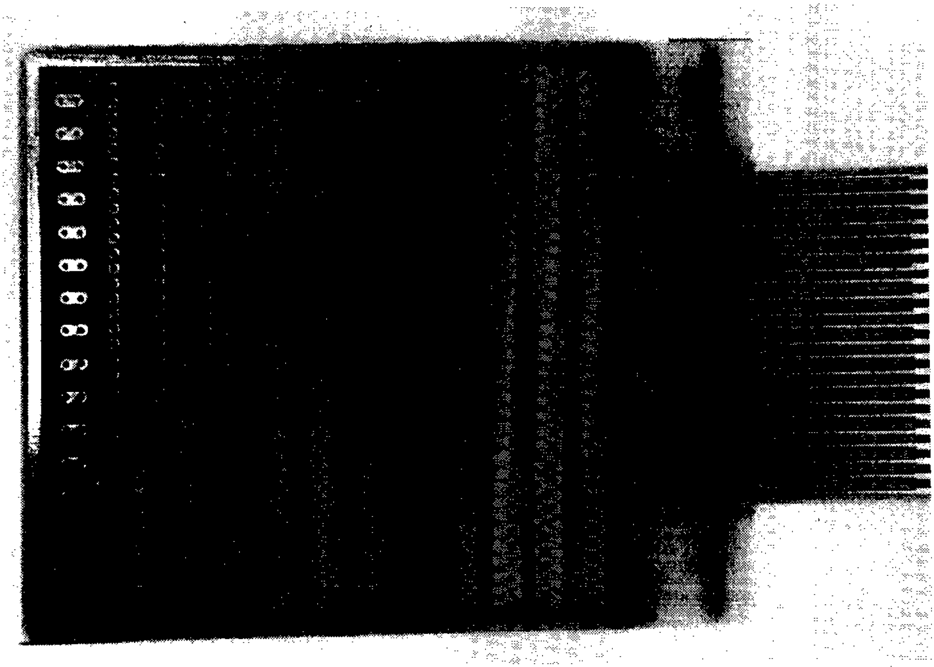


Sockel auf eine Experimentierplatine aufgesteckt (links im Bild)

Die Experimentierplatine können Sie auch als Stecker verwenden, indem Sie die Platine einstecken und die Drähte entsprechend oben und unten an der Platine anlöten. Bitte beachten Sie den Fehler im "Commodore 64 Micro Computer Handbuch", Seite 142.

Richtige Pin-Belegung von hinten gesehen:

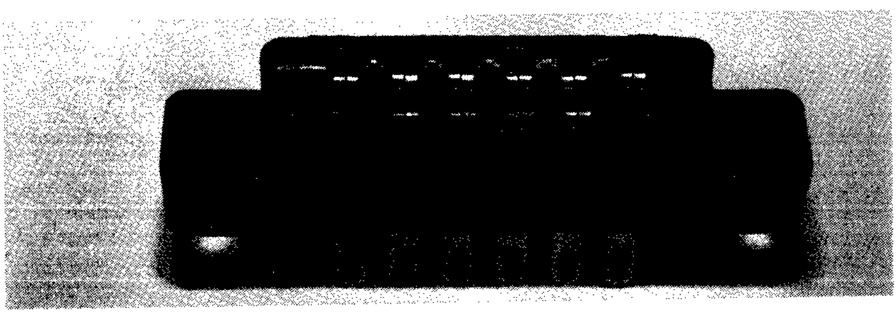




Hier die Platine Nr. 4970, DM 39,–

Diese Platine kann auch als Stecker für Modul-Steckplatz verwendet werden.

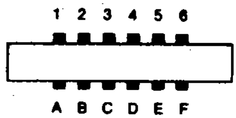
Anschlußstecker für den Cassettenport



Best.-Nr. 4996, DM 9,80

Wer beim Experimentieren am Commodore 64 noch mehr Leistung braucht, kann noch den Cassettenport anzapfen. An Pin 2 (+5V) und

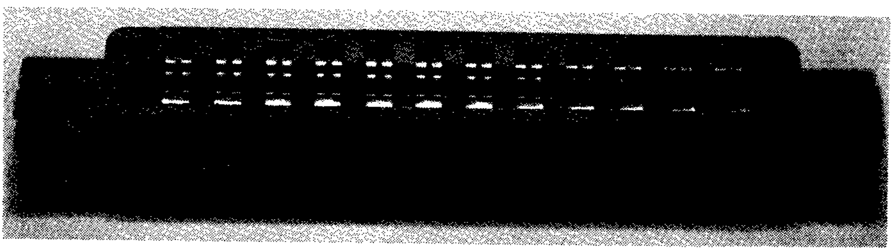
Pin 1 (GND) können noch Lasten bis zu ca. 0,5A angeschlossen werden.



Pin-Belegung von hinten gesehen

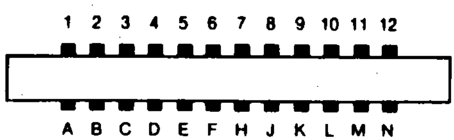
USER-Port (Anwenderport) Stecker

Der 22-polige USER-Port Stecker sieht wie folgt aus:



USER-Port Stecker, 22 polig, Best.-Nr. 4847, DM 19,80

Er wird von außen auf den männlichen Platinenstecker hinten links am C-64 aufgesteckt. Die Verbindungsdrähte werden dann an die Lötfahnen angelötet.

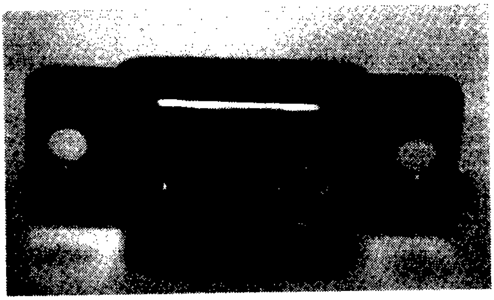


Anschlußbild

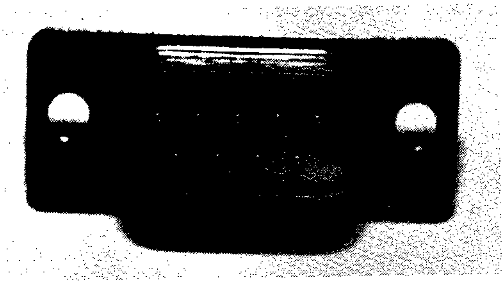
Joystick-Port-Stecker

Wer einen eigenen Joystick anschließen, oder auf den Lichtgriffel oder A/D-Wandler-Anschluß zugreifen will, braucht einen neunpoligen

Stecker (Weibchen), wie im folgenden Bild dargestellt.



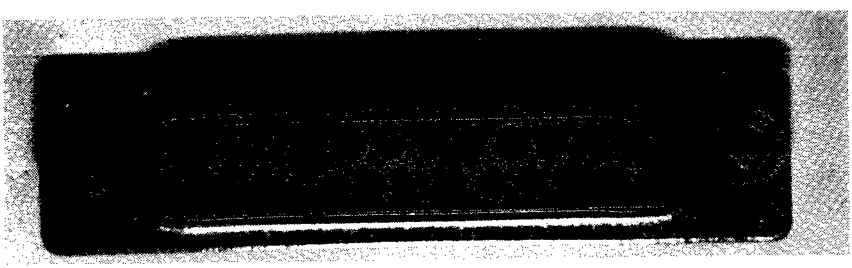
Joystick-Port-Stecker, 9polig (Weibchen), Best.-Nr. 7040, DM 9,80 /Stück



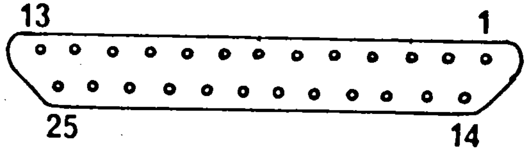
Der zugehörige Stiftstecker (Männchen)

25plige RS-232 Stecker

Der nachfolgend beschriebene Stecker wird meistens im Zusammenhang mit RS-232-Schnittstellen verwendet. Am Computer und am Drucker befindet sich meist ein Weibchen (siehe folgendes Bild).

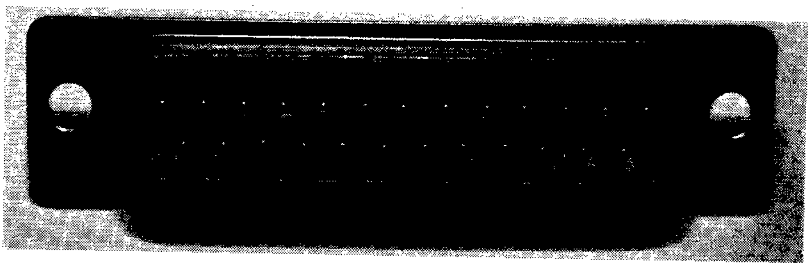


25poliger RS-232 Stecker (Buchse)

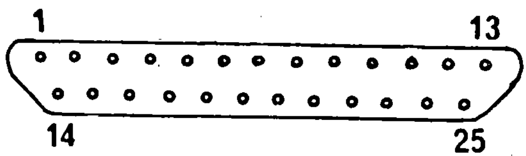


Anschlußbild von hinten gesehen

Wer sich selbst ein Druckerkabel anfertigen will, braucht dazu das dazugehörige Männchen. Achtung: Hier erfolgt die Nummerierung in entgegengesetzter Richtung.



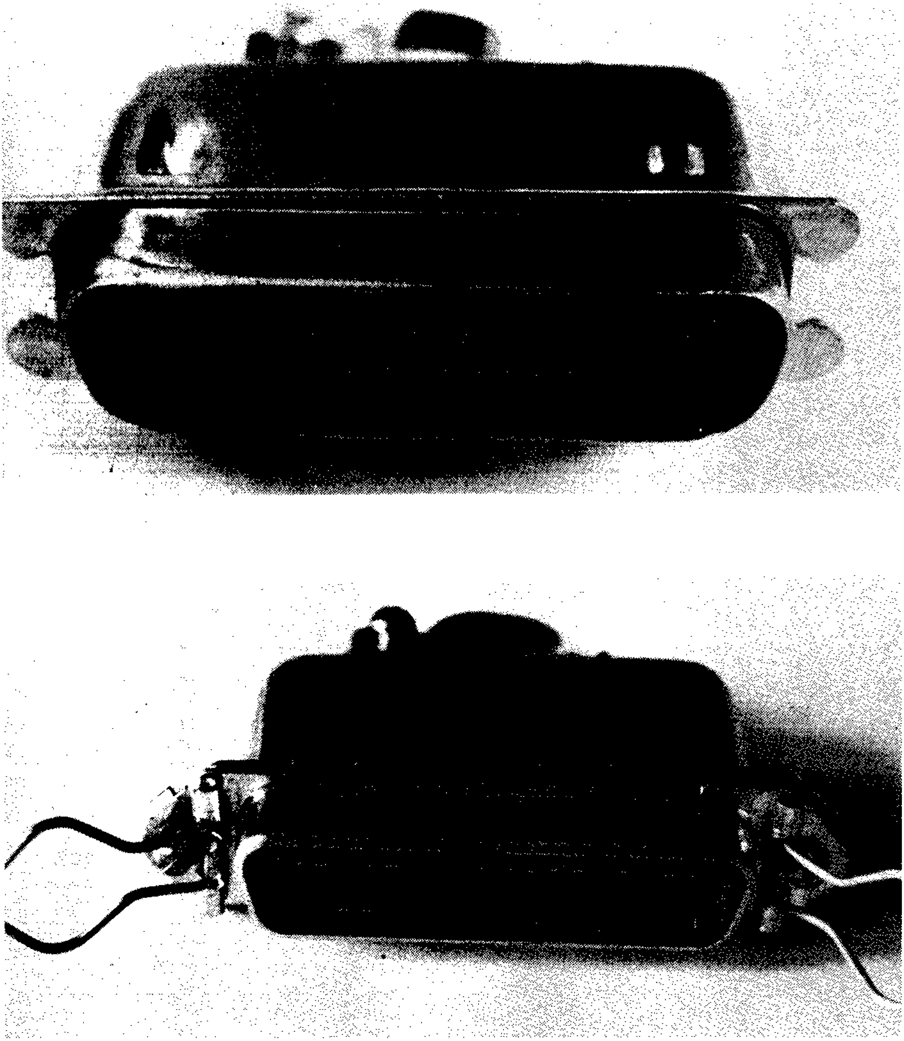
RS-232 Stecker (Stift) 25polig



Anschlüsse von hinten gesehen

Wer einen Typenraddrucker mit serieller Schnittstelle über den USER-Port anschließen will, braucht einen solchen 25poligen Stecker mit Stiften und einen USER-Port Stecker Nr. 4747. Wie man die Verbindungen herstellt, erfahren Sie im Anhang A.

Zum Schluß noch ein anderer, weit verbreiteter Stecker. Er wird bei nahezu allen Centronic kompatiblen parallelen Schnittstellen und beim IEC-Bus verwendet. Es handelt sich hierbei um den 36poligen Stecker, wie er im folgenden Bild dargestellt ist.



Anschlußstecker für parallele Drucker (EPSON, Star, Centronics, Okidata usw)

Dieser Stecker ist beim Fachhandel erhältlich.

Notizen

F Verbindung des TRS-80 Model-100 mit C-64 und Blitztext

Verbindung des TRS-80 Model-100 mit dem C-64 und Bliztext

Der Terminalmodus, mit dem BLIZTEXT als erster Wortprozessor weltweit ausgerüstet ist, erlaubt es, serielle Daten mit 300 Baud, halbduplex im Dreileitungsverfahren zu senden und zu empfangen.



Model 100

wird F3 gedrückt für "UPLOAD". Da zu sendene Testfile muß als DO File gespeichert sein.
Nun geben wir den Filenamen ein und die Breite, nach dem ein Linefeed gesendet werden soll (z. B. 40).

Jetzt sendet Model 100 und wir müssen den Text auf dem Bildschirm des C-64 verfolgen.

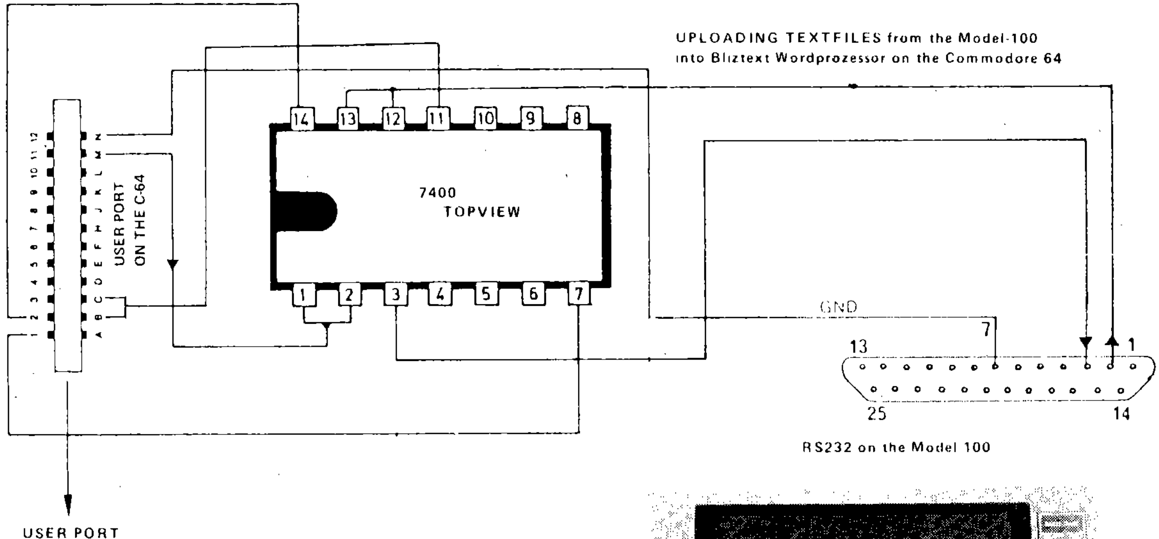
Wenn die Übertragung beendet ist, drücken wir F8 und trennen das Model 100 mit "Disconnected" ab. Am C-64 geben wir ⟨SHIFT⟩ ⟨F1⟩ ein, und kehren zurück in den Wortprozessor. Jetzt steht uns der Text zur weiteren Verarbeitung zur Verfügung.

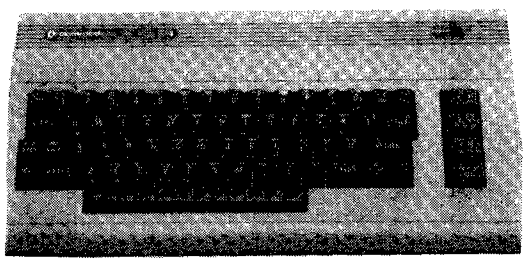
Schlußbemerkung:

Die vorher beschriebene Datenübertragung wurde getestet und arbeitet. Als Beispiel wurde in den USA ein Telex aus Übersee mit dem Model 100 empfangen und hier bei uns vom Model 100 in den C-64 geladen.

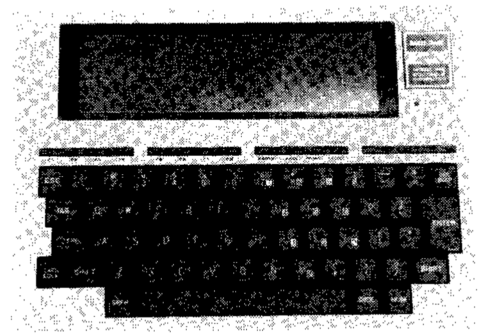
Ähnliches können Sie auch mit Text, Adressen und BASIC-Programmen machen.

UPLOADING TEXTFILES from the Model-100
into Bliztext Wordprozessor on the Commodore 64





Commodore 64 with BLIZTEXT



TANDY Model 100

Bezeichnungen:

n, n1	16 bit Zahlen mit Vorzeichen
d, d1	32 bit Zahlen mit Vorzeichen
u, u1	16 bit Zahl ohne Vorzeichen
addr	Adresse
b	8 bit Byte
c	7 bit ASCII-Zeichen
f	Bool'sche Zahl

Bei der Angabe der Veränderungen des Stapels ist links der Zustand des Stapels vor dem Aufruf des Wortes, rechts der Zustand des Stapels nach dem Aufruf des Wortes angegeben. Das in dieser Bezeichnung am weitesten rechts stehende Element ist oberstes Element des Stapels.

STAPELÄNDERUNGEN:

DUP	(n --> n n)	Verdoppeln der obersten Zahl
DROP	(n -->)	Löschen der obersten Zahl
SWAP	(n1 n2 --> n2 n1)	Vertauschen der beiden obersten Zahlen
OVER	(n1 n2 --> n1 n2 n1)	Zweite Zahl des Stapels im TOS kopieren
ROT	(n1 n2 n3 --> n2 n3 n1)	Zyklisches Vertauschen der obersten drei Zahlen
–DUP	(n --> n ?)	Verdoppeln der Zahl, nur wenn sie nicht Null ist
>R	(n -->)	Oberste Zahl auf den Return-Stapel schreiben
R>	(--> n)	Oberste Zahl des Return-Stapels auf den Stapel holen
R	(--> n)	Oberste Zahl des Return-Stapels auf den Stapel kopieren

ARITHMETISCHE UND LOGISCHE BEFEHLE:

+	(n1 n2 --> sum)	Addieren
D+	(d1 d2 --> sum)	Addieren doppelt langer Zahlen
–	(n1 n2 --> diff)	Subtrahieren (n1 – n2)
*	(n1 n2 --> prod)	Multiplizieren
/	(n1 n2 --> quot)	Dividieren (n1 / n2)
MOD	(n1 n2 --> rem)	Rest der Division (Modulo)
/MOD	(n1 n2 --> rem quot)	Quotient und Rest der Division

*/MOD	(n1 n2 n3 --> rem quot)	Berechnung von n1 * n2 / n3 mit doppelt langen Zwischenergebnis
*/	(n1 n2 n3 --> quot)	Wie */MOD, nur Quotient auf dem Stapel
MAX	(n1 n2 --> max)	Max ist die größere der beiden Zahlen n1, n2
MIN	(n1 n2 --> min)	Min ist die kleiner der beiden Zahlen n1, n2
ABS	(n --> absolute)	Absolutwert einer 16 bit Zahl
DABS	(d --> absolute)	Absolutwert einer 32 bit Zahl
MINUS	(n --> −n)	16 bit Zahl wird negativ
DMINUS	(d --> −d)	32 bit Zahl wird negativ
AND	(n1 n2 --> and)	Logisches UND, bitweise
OR	(n1 n2 --> or)	Logisches ODER, bitweise
XOR	(n1 n2 --> xor)	Logisches EXCLUSIV ODER, bitweise

STEUERWORTE:

DO . . . LOOP		
DO	(end+1 start -->)	Setzt Schleifenparameter
LOOP	(-->)	Erhöht Zählparameter um Eins
I	(--> n)	Holt Zählparameter auf Stapel
LEAVE	(-->)	Verlässt die Schleife beim nächsten LOOP oder +LOOP
DO . . . +LOOP		Wie DO . . . LOOP, aber addiert n zu Zählparameter
+LOOP	(n -->)	
IF . . . ENDIF	(f -->)	Programm zwischen IF und ENDIF (THEN) wird ausgeführt,
IF . . . THEN	(f -->)	wenn der Wert im TOS ungleich Null ist
IF . . . ELSE . . . ENDIF	(f -->)	Programm zwischen IF und ELSE wird ausgeführt, wenn Wert im TOS ungleich Null ist, sonst wird Programm zwischen ELSE und ENDIF (THEN) ausgeführt.
IF . . . ELSE . . . THEN		
BEGIN . . . UNTIL	(f -->)	Programm zwischen BEGIN und UNTIL (END) wird solange aus-
BEGIN . . . END	(f -->)	geführt, bis im RPS vor UNTIL (END) ein Wort ungleich Null ist
BEGIN . . . WHILE		Solange der TOS vor WHILE ungleich Null ist, wird Programm
. . . REPEAT	(f -->)	zwischen BEGIN und REPEAT ausgeführt. Mit TOS gleich Null vor WHILE wird Schleife verlassen.

EIN- AUSGABEFORMATIERUNG:

NUMBER	(addr --> d)	Setzt eine Zeichenkette ab Adresse addr in eine doppelt lange Zahl um
<#	(-->)	Beginn der Zeichenformatierung
#	(d --> d)	Wandle eine Zahlenstelle in ASCII-Zeichen
#S	(d --> 0 0)	Wandle restliche Stelle in ASCII-Zeichen
SIGN	(n d --> d)	Setze Vorzeichen ein
#>	(d --> addr u)	Beende Formatierung
HOLD	(c -->)	Füge ASCII-Zeichen c in Zeichenkette ein

WÖRTERBUCH:

CONTEXT	(--> addr)	Bringt Adresse des CONTEXT Wörterbuches
CURRENT	(--> addr)	Bringt Adresse des CURRENT Wörterbuches
FORTH	(-->)	Setzt CONTEXT und CURRENT auf FORTH Wörterbuch
DEFINITIONS	(-->)	Setzt CURRENT auf CONTEXT
VOCABULARY xxx	(-->)	Erzeugt Wörterbuch xxx
VLIST	(-->)	Listet Inhalt des Wörterbuches
FENCE	(--> addr)	Systemvariable, enthält die Adresse des obersten geschützen Eintrags im Wörterbuch

SYSTEM WORTE:

(	(-->)	Beginn eines Kommentars, Kommentarende ist)
FORGET xxx	(-->)	Löschen eines und aller späteren Einträge im Wörterbuch
ABORT	(-->)	Fehlerabbruch
'xxx	(--> addr)	Suche Codefeldadresse von xxx
HERE	(--> addr)	Adresse der nächsten freien Stelle im Wörterbuch
PAD	(--> addr)	Adresse von PAD
IN	(--> n)	Offset zum Eingabespeicher

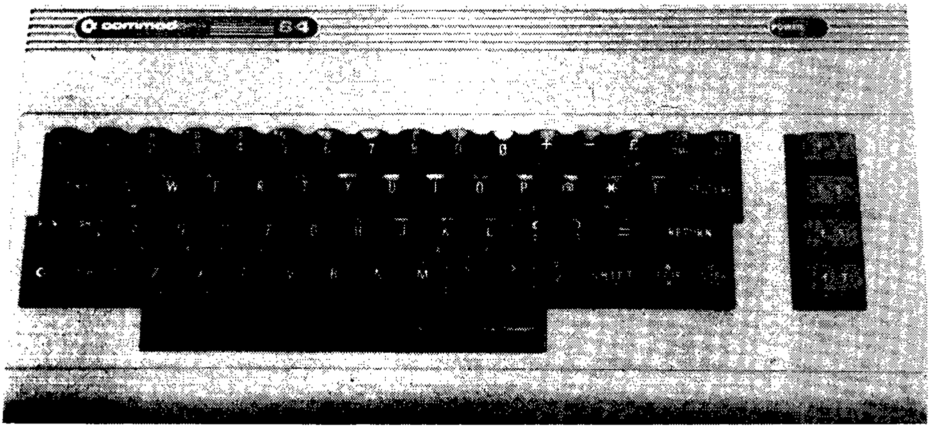
SP@	(--> addr)	Adresse des TOS
ALLOT	(n -->)	n Bytes im Wörterbuch reservieren
,	(n -->)	Speichern von n an Adresse HERE
C,	(b -->)	Speichern des Byte b an Adresse HERE

DISKETTENSPEICHERUNG:

LIST	(n -->)	Textfeld (Screen) n wird ausgegeben
LOAD	(n -->)	Textfeld (Screen) n wird compiliert
BLOCK	(n --> addr)	Diskettenblock n wird eingelesen, Anfangsadresse im Speicher auf Stapel
FLUSH	(-->)	Geänderte Blöcke werden auf Diskette geschrieben.
UPDATE	(-->)	Augenblicklichen Block als geändert markieren
EMPTY-BUFFERS	(-->)	Alle Blöcke im Speicher löschen
BLK	(--> addr)	Variable, enthält augenblickliche Blocknummer
B/BUF	(--> n)	Constante, Blockgröße in Byte
SCR	(--> addr)	Variable, enthält augenblickliche Textfeldnummer

NEUE BOCHER

für den



von HOFACKER

More on the Sixtyfour (64)

Eine echte Sensation weltweit! Ein Buch voll gestopft mit wichtigen und nützlichen Maschinenprogrammen. Rekursive Routinen, Ein-/ Ausgabe von Text, Echtzeituhr, RS-232 und Centronics Treiber, A/D-Wandler am Joystickport und vieles andere mehr (englisch).

Best.-Nr. 183 — **39,– DM**

Programmieren in Maschinensprache mit dem Commodore-64

Einführung in die 6502 Maschinensprache auf dem C-64. Sehr viele Beispiele. Der gesamte Befehlssatz. Ein ausgezeichnetes Buch.

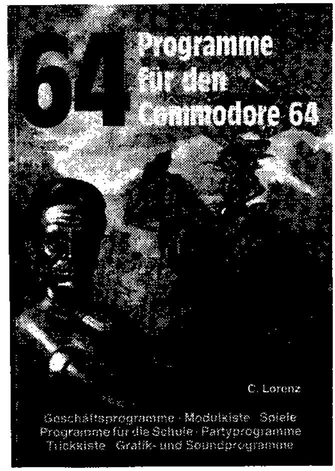
Best.-Nr. 124 — **19,80 DM**

The Great Book of Games Vol. I
Games for the Commodore-64

Wie programmiere ich eigene Spiele. Einführung in Graphik, Tonausgabe und Bewegung mit sehr vielen Beispielen (englisch).

Best.-Nr. 182 — **29,80 DM**





Beherrschen Sie Ihren Commodore 64

Auf über 150 Seiten findet der C-64 Besitzer die Zusatzinformationen, die er noch zusätzlich zum mitgelieferten Handbuch benötigt. Grundlagen, ein Blick ins Innere von BASIC, einfacher Programmschutz, Listschutz-Tricks und Programmiertips. Einführung in Dateien auf Cassette und Diskette. Ton und Grafik, nützliche Hilfsprogramme und hochauflösende Grafik.

Best.-Nr. 147 **19,80 DM**

64 Programme für den Commodore-64

Diese Programmsammlung braucht jeder C-64 Besitzer. Sie finden darin Geschäftsprogramme für den C-64, nützliche BASIC-Programme (Modulkiste), Programme für den Schüler und Studenten (Mathematik und Statistik, Spiele, Partyprogramme, verschiedene Programme mit Grafik und Sound, sowie Tips und Tricks.

Best.-Nr. 145 **39,– DM**

Mehr als 29 Programme für den Commodore-64

Eine sehr interessante Programmsammlung für den Commodore-64. Viele wertvolle Programme wie ein Luxus, Sprite Editor, Stichwortdatei, mathematische Programme, Lagerverwaltung, Adressenverwaltung und Fakturierung u. v. m.

Best.-Nr. 187 **29,80 DM**

Hardware Erweiterungen für den Commodore-64

Eine ausführliches Anleitungsbuch für jeden der seinen C-64 für Steuerungen und Hardware Experimente verwenden will. 4 Joysticks am C-64, A/D-Wandler, Druckeranschluß u. v. a. (englisch).

Best.-Nr. 146 **39,– DM**

Fordern Sie unseren Katalog an !

150 Seiten vollgepackt mit neuen Büchern für Elektronik und Microcomputer.



Software für:

- COMMODORE-64
- VC-20
- PET/CBM
- ATARI 400/800
- SINCLAIR
- TRS-80
- GENIE
- APPLE II
- OSBORNE

Heute noch bestellen !

2,– DM in Briefmarken oder Vorkasse auf Postscheckkonto München 15 994–807.

Ing. W. Hofacker GmbH
Tegernseer Straße 18
D- 8150 Holzkirchen

Telefon (0 80 24) 73 31, Telex: 52 69 73

MACROFIRE

Der Macro-Assembler für den Commodore-64 von HOFACKER

Ein extrem leistungsfähiges Werkzeug für den Programmentwickler. Sehr hohe Geschwindigkeit. Übersetzt ca. 10K Quelltext in 3 – 4 Sekunden.

Dieser Assembler wurde weltweit bereits mehrere hundert Mal verkauft und wird bei führenden US-Softwarefimren in der System- und Anwendersoftwareentwicklung eingesetzt.

Das Paket besteht aus einem Editor (ähnlich unserem Wortprozessor BLIZTEXT), dem Assembler und einem komfortablen Maschinensprachen-Monitor.
Das Paket ist integriert. Sie erreichen den Assembler oder den Monitor vom Editor aus über einen Tastendruck. Es wird in drei Durchgängen übersetzt. Bis zu 1000 Labels können vereinbart werden. Volle Include-Funktion auf Cassette oder Diskette. Sehr komfortable Anwenderführung, ca. 45 Kommandos im Editor. Alle Commodore-Disketten und Cassetten I/O-Befehle werden unterstützt. Unseren Kenntnissen nach einer der leistungsfähigsten Editor/Assembler-Pakete weltweit.

Lieferung erfolgt vorerst mit englischem Manual. Ein deutsches Handbuch wird nachgereicht. Eine einmalige, an die Person gebundene Benutzerlizenz kostet 199,– incl. MwSt.

Best.-Nr. 4964 **199,– DM**

BLIZTEXT TM

Nach unserem Wissen einer der besten Wortprozessoren für den C-64

Ab sofort liefert die Firma Ing. W. Hofacker GmbH in Holzkirchen ein sehr leistungsfähiges Textverarbeitungsprogramm für den Commodore-64.
Dieses neue Programm erlaubt es dem Anwender, ein Textsystem zu einem noch bis heute undenkbar niedrigen Preis zusammenzustellen.

Erforderliche Hardware: (Minimum)

- 1 Commodore-64
- 1 Cassettenrecorder oder Diskette 1541
- 1 Drucker, ähnlich VC1525
- 1 Bildschirmmonitor oder Fernsehgerät

Das Textverarbeitungsprogramm besteht aus drei Teilen, dem Editor, dem Formatter und einem Terminalprogramm.
Alle drei Systemelemente sind zu einem ergonomischen und leicht zu bedienenden Softwarepaket integriert.
Das Programm bietet dem Anwender praktisch alles, was er von einer professionellen Textverarbeitung erwarten kann.
Hier kurz eine Zusammenfassung der wichtigsten Eigenschaften:

1.) Voll bildschirmorientiert, horizontales und vertikales Scrolling bis zu 255 Zeichen / Zeile
2.) Statusanzeige für Zeile und Zeichen in der Zeile
3.) 27K RAM frei für Text im Hauptspeicher plus 4K Kopierpuffer
4.) Echte Include-Funktion auf Cassette und Diskette. Dies bedeutet, daß Sie Textfiles auf Diskette in den momentanen Text einbinden können.
5.) Linker und rechter Randausgleich sowie Zentrierung sind natürlich selbstverständlich
6.) Ca. 30 Kommandos stehen im Editor zur Verfügung, weitere 20 Befehle zur Textformattierung. Alle Commodore-Cassetten und Disk-I/O-Befehle werden voll unterstützt.
7.) Verbesserte Fehlerkanal-Leseoperation, Directory-Abfrage durch 2 Tastenbetätigungen
8.) Groß- und Kleinschreibung auf praktisch alle Drucker (RS232, IEEE, IEC)
9.) Dynamische Formattierung durch direkt in den Text eingefügte Kommandos
10.) Komfortable Fehlermeldungen
11.) Druckersteuerzeichen können individuell in den Text eingebaut werden. Somit können Sie auf fast allen Druckern unterstreichen, Breitschrift, Fettschrift usw.
12.) **Eingebauter Terminalmodus** – Dies hat es bis jetzt weltweit noch nicht gegeben. Ein Terminal kann zum Senden und Empfangen von elektronischer Post simuliert werden. Der Modus kann auch zur Verwendung von Rechnerkopplungen mit anderen Personalcomputern verwendet werden.

Ein Produkt, für das wir Ihnen praktisch eine Zufriedenheitsgarantie geben könnten.
Ein Manual mit ca. 70 Seiten in englischer Sprache ist verfügbar. Ein ausführliches, deutsches Handbuch ist in Arbeit.
Verfügbar auf Cassette oder Diskette.
Der Wortprozessor "BLIZTEXT" kann sofort bestellt werden. Der Preis beträgt **DM 199,00** incl. Mwst. und englischem Handbuch. Ein deutsches Handbuch wird nachgeliefert.
Das im Preis enthaltene Handbuch wurde mit BLIZTEXT selbst erstellt und kann für **DM 49,00** alleine bezogen werden.
Der Kaufpreis wird später angerechnet.

Ing. W. Hofacker GmbH, Tegernseer Str. 18, D-8150 Holzkirchen, Tel.: ☎ 8024 / 73 31, Tlx.: 52 69 73

Weitere interessante Bücher von Hofacker:

Best.-Nr.	Titel	Preis/DM
	Bücher in deutscher Sprache aus dem Hofacker-Verlag	
1	Transistor Berechnungs- und Bauanleitungsbuch – 1	29,80
2	Transistor Berechnungs- und Bauanleitungsbuch – 2	19,80
3	Elektronik im Auto	9,80
4	IC-Handbuch, TTL, CMOS, Linear	19,80
5	IC-Datenbuch, TTL, CMOS, Linear	9,80
6	IC-Schaltungen, TTL, CMOS, Linear	19,80
7	Elektronik Schaltungen	19,80
8	IC-Bauanleitungs-Handbuch	19,80
9	Feldeffekttransistoren	9,80
10	Elektronik und Radio	19,80
11	IC-NF Verstärker (i. V.)	9,80
12	Beispiele integrierter Schaltungen (BIS)	19,80
13	HEH, Hobby Elektronik Handbuch	9,80
15	Optoelektronik Handbuch	19,80
16	CMOS Teil 1, Einführung, Entwurf, Schaltbeispiele	19,80
17	CMOS Teil 2, Entwurf und Schaltbeispiele	19,80
18	CMOS Teil 3, Entwurf und Schaltbeispiele	19,80
19	IC-Experimentier Handbuch	19,80
20	Operationsverstärker	19,80
21	Digitaltechnik Grundkurs	19,80
22	Mikroprozessoren, Eigenschaften und Aufbau	19,80
23	Elektronik Grundkurs, Kurzlehrgang Elektronik	9,80
24	Progr. in Maschinensprache mit Z-80, Band II	29,80
25	68000 Microcomputer Einführung (i. V.)	39,00
26	Mikroprozessor, Teil 2	19,80
27	BASIC-M Anwender-HB f. 6800/6809/68000 (Motorola)	29,80
28	Lexikon + Wörterbuch f. Elektr. u. Mikroprozessor	29,80
29	Mikrocomputer Datenbuch	49,80
30	Floppy Disk Selbstbau-Handbuch (i. V.)	49,00
31	57 Programme in BASIC	39,00
33	Microcomputer Programmierbeispiele	19,80
34	TINY-BASIC Handbuch	19,80
35	Der freundliche Computer	29,80
103	Oszillographen-Handbuch	19,80
108	Rund um den Spectrum (Progr., Tips und Tricks)	29,80
109	6502 Microcomputer Programmierung	29,80
110	Programmierhandbuch für PET	29,80
111	Programmieren mit TRS-80 (Video Genie)	29,80
112	PASCAL-Programmier-Handbuch	29,80
113	BASIC-Programmier-Handbuch	19,80
114	Der Microcomputer im Kleinbetrieb	39,80
115	6809 Programmier Handbuch (i. V.)	49,00
116	Einführung 16-Bit Microcomputer	29,80
117	FORTRAN für Heimcomputer	19,80
118	Programmieren in Maschinensprache mit dem 6502	49,00
119	Programmieren in Maschinensprache (Z80) Band I	39,00
120	Anwenderprogramme für TRS-80 u. Video Genie	29,80
121	Microsoft BASIC-Handbuch	29,80
122	BASIC für Fortgeschrittene	39,00
123	IEC-Bus Handbuch	19,80
124	Programmieren i. Ma.-Sprache mit Commodore-64	29,80
127	Einführung i. d. Microcmputer-Progr. mit 6800	49,00
128	Programmieren mit dem CBM	29,80
130	Programmierbeispiele für CBM	19,80
132	CP/M-Handbuch	19,80

Best.-Nr.	Titel	Prei
133	Handbuch für MS/DOS (i. V.)	
137	FORTH Handbuch	
139	BASIC für blutige Laien	
140	Progr. i. BASIC u. Maschinencode mit dem ZX81	
141	Programme f. VC-20 (Spiele, Utilities, Erweiterungen	
143	35 Programme für den ZX81	
144	33 Programme für den ZX-Spectrum	
145	64 Programme für den Commodore 64	
146	Hardware-Erweiterungen für den C-64 (i. V	
147	Beherrschen Sie Ihren Commodore 64	
148	Programmierhandbuch für SHARP	
149	Programme für TI 99/4A	
175	Astrologie auf dem ATARI 800	
8029	Z-80 Assembler-Handbuch	
	Bücher in englischer Sprache	
	1. Von ELCOMP Publishing, Inc., Los Angeles, C	
150	Care and Feeding of the Commodore PET	
151	8K Microsoft BASIC Reference Manual	
152	Expansion Handbook for 6502 and 6800	
154	Complex Sound Generation using the SN76477	
156	Small Business Programs	
158	The Second Book of Ohio Scientific	
159	The Third Book of Ohio Scientific	
160	The Fourth Book of Ohio Scientific	
161	The Fifth Book of Ohio Scientific	
162	ATARI Games in BASIC	
163	The Peripheral Handbook (i. V.)	
164	ATARI-BASIC Learning by Using	
166	Programming in 6502 Machinelanguage PET/CBM	
169	How to Progr. your ATARI in 6502 Machinelanguage	
170	FORTH on the ATARI – Learning by Using	
171	See the Future with your ATARI (Astrology)	
172	Hackerbook I (Tricks + Tips for your ATARI)	
173	PD-Program Descriptions (ATARI)	
174	ZX-81/TIMEX Progr. i. BASIC a. Machine Lang.	
176	Programs + Tricks for VIC's	
177	CP/M – MBASIC and the OSBORNE	
178	The APPLE in Your Hand	
182	The Great Book of Games Vol. I - Games f. the C-64	
183	More on the Sixtyfour	
	Riesenprogrammsammlung in BASIC	
8048	BASIC Software Vol. VI	
8049	BASIC Software Vol. VII	
8050	BASIC Software Vol. I	
8051	BASIC Software Vol. II	
8052	BASIC Software Vol. III	
8053	BASIC Software Vol. IV	
8054	BASIC Software Vol. V	

Der Hofacker Verlag produziert und vertreibt neben ein großen Auswahl an Fachbuchern für Elektronik und computertechnik noch:

- Leerplatinen und Bauanleitungen fur Zusatzeinrichtunge Ihren Personalcomputer, sowie
- Programme (Software) fur die bedeutenden Personalcom

(i. V. bedeutet: Buch ist in Vorbereitung)

HOFACKER

HOLZKIRCHEN SINGAPORE LOS ANGE

ISBN 3-88963-146-0